I2NSF
Internet-Draft
Intended status: Standards Track
Expires: February 6, 2020

R. Marin-Lopez
G. Lopez-Millan
University of Murcia
F. Pereniguez-Garcia
University Defense Center

August 5, 2019

Software-Defined Networking (SDN)-based IPsec Flow Protection
draft-ietf-i2nsf-sdn-ipsec-flow-protection-07

Abstract

This document describes how providing IPsec-based flow protection by means of a Software-Defined Network (SDN) controller (aka. Security Controller) and establishes the requirements to support this service. It considers two main well-known scenarios in IPsec: (i) gateway-to-gateway and (ii) host-to-host. The SDN-based service described in this document allows the distribution and monitoring of IPsec information from a Security Controller to one or several flow-based Network Security Function (NSF). The NSFs implement IPsec to protect data traffic between network resources.

The document focuses on the NSF Facing Interface by providing models for configuration and state data required to allow the Security Controller to configure the IPsec databases (SPD, SAD, PAD) and IKEv2 to establish Security Associations with a reduced intervention of the network administrator.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Software-Defined Networking (SDN) is an architecture that enables users to directly program, orchestrate, control and manage network resources through software. The SDN paradigm relocates the control of network resources to a dedicated network element, namely SDN Controller. The SDN controller (or Security Controller in the context of this document) manages and configures the distributed network resources and provides an abstracted view of the network resources to the SDN applications. The SDN application can customize and automate the operations (including management) of the abstracted network resources in a programmable manner via this interface [RFC7149] [ITU-T.Y.3300] [ONF-SDN-Architecture] [ONF-OpenFlow].

Recently, several network scenarios are considering a centralized way of managing different security aspects. For example, Software-Defined WANs (SD-WAN), an SDN extension providing a software abstraction to create secure network overlays over traditional WAN and branch networks. SD-WAN is based on IPsec as underlying security protocol and aims to provide flexible, automated, fast deployment and on-demand security network services such as IPsec SA management from a centralized point.

Therefore, with the growth of SDN-based scenarios where network resources are deployed in an autonomous manner, a mechanism to manage IPsec SAs according to the SDN architecture becomes more relevant. Thus, the SDN-based service described in this document will autonomously deal with IPsec SAs management following the SDN paradigm.

IPsec architecture [RFC4301] defines clear separation between the processing to provide security services to IP packets and the key management procedures to establish the IPsec Security Associations. In this document, we define a service where the key management procedures can be carried by an external and centralized entity: the Security Controller.

First, this document exposes the requirements to support the protection of data flows using IPsec [RFC4301]. We have considered two general cases:

1) IKE case. The Network Security Function (NSF) implements the Internet Key Exchange (IKE) protocol and the IPsec databases: the Security Policy Database (SPD), the Security Association Database (SAD) and the Peer Authorization Database (PAD). The Security Controller is in charge of provisioning the NSF with the required information to IKE, the SPD and the PAD.

2) IKE-less case. The NSF only implements the IPsec databases (no IKE implementation). The Security Controller will provide the required parameters to create valid entries in the SPD and the SAD into the NSF. Therefore, the NSF will have only support for IPsec while automated key management functionality is moved to the Security Controller.

In both cases, an interface/protocol is required to carry out this provisioning in a secure manner between the Security Controller and the NSF. In particular, IKE case requires the provision of SPD and PAD entries, the IKE credential and information related with the IKE negotiation (e.g. IKE_SA_INIT). IKE-less case requires the management of SPD and SAD entries. Based on YANG models in [netconf-vpn] and [I-D.tran-ipsecme-yang], RFC 4301 [RFC4301] and RFC 7296 [RFC7296], this document defines the required interfaces with a YANG model for configuration and state data for IKE, PAD, SPD and SAD (see Appendix A, Appendix B and Appendix C). Examples of the usage of these models can found in Appendix D, Appendix E and Appendix F.

This document considers two typical scenarios to manage autonomously IPsec SAs: gateway-to-gateway and host-to-host [RFC6071]. In these cases, hosts, gateways or both may act as NSFs. Finally, it also discusses the situation where two NSFs are under the control of two different Security Controllers. The analysis of the host-to-gateway (roadwarrior) scenario is out of scope of this document.

Finally, this work pays attention to the challenge "Lack of Mechanism for Dynamic Key Distribution to NSFs" defined in [RFC8192] in the particular case of the establishment and management of IPsec SAs. In fact,this I-D could be considered as a proper use case for this particular challenge in [RFC8192].

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC

2119 [RFC2119]. When these words appear in lower case, they have their natural language meaning.

3. Terminology

This document uses the terminology described in [RFC7149], [RFC4301], [ITU-T.Y.3300], [ONF-SDN-Architecture], [ONF-OpenFlow], [ITU-T.X.1252], [ITU-T.X.800] and [I-D.ietf-i2nsf-terminology]. In addition, the following terms are defined below:

- Software-Defined Networking. A set of techniques enabling to directly program, orchestrate, control, and manage network resources, which facilitates the design, delivery and operation of network services in a dynamic and scalable manner [ITU-T.Y.3300].

- Flow/Data Flow. Set of network packets sharing a set of characteristics, for example IP dst/src values or QoS parameters.

- Security Controller. An entity that contains control plane functions to manage and facilitate information sharing, as well as execute security functions. In the context of this document, it provides IPsec management information.

- Network Security Function (NSF). Software that provides a set of security-related services.

- Flow-based NSF. A NSF that inspects network flows according to a set of policies intended for enforcing security properties. The NSFs considered in this document fall into this classification.

- Flow-based Protection Policy. The set of rules defining the conditions under which a data flow MUST be protected with IPsec, and the rules that MUST be applied to the specific flow.

- Internet Key Exchange (IKE) v2. Protocol to establish IPsec Security Associations (SAs). It requires information about the required authentication method (i.e. raw RSA/ECDSA keys or X.509 certificates), Diffie-Hellman (DH) groups, IPsec SAs parameters and algorithms for IKE SA negotiation, etc.

- Security Policy Database (SPD). It includes information about IPsec policies direction (in, out), local and remote addresses (traffic selectors information), inbound and outboud IPsec SAs, etc.

- Security Associations Database (SAD). It includes information about IPsec SAs, such as SPI, destination addresses,

authentication and encryption algorithms and keys to protect IP flows.

- Peer Authorization Database (PAD). It provides the link between the SPD and a security association management protocol. It is used when the NSF deploys IKE implementation (IKE case).

4. Objectives

- To describe the architecture for the SDN-based IPsec management, which implements a security service to allow the establishment and management of IPsec security associations from a central point, in order to protect specific data flows.

- To define the interfaces required to manage and monitor the IPsec Security Associations (SA) in the NSF from a Security Controller. YANG models are defined for configuration and state data for IPsec management.

5. SDN-based IPsec management description

As mentioned in Section 1, two cases are considered, depending on whether the NSF ships an IKEv2 implementation or not: IKE case and IKE-less case.

5.1. IKE case: IKE/IPsec in the NSF

In this case the NSF ships an IKEv2 implementation besides the IPsec support. The Security Controller is in charge of managing and applying IPsec connection information (determining which nodes need to start an IKE/IPsec session, deriving and delivering IKE Credentials such as a pre-shared key, certificates, etc.), and applying other IKE configuration parameters (e.g. cryptographic algorithms for establishing an IKE SA) to the NSF for the IKE negotiation.

With these entries, the IKEv2 implementation can operate to establish the IPsec SAs. The application (administrator) establishes the IPsec requirements and information about the end points information (through the Client Facing Interface, [RFC8192]), and the Security Controller translates these requirements into IKE, SPD and PAD entries that will be installed into the NSF (through the NSF Facing Interface). With that information, the NSF can just run IKEv2 to establish the required IPsec SA (when the data flow needs protection). Figure 1 shows the different layers and corresponding functionality.

```
               +-------------------------------------------+
               |IPsec Management/Orchestration Application | Client or
               |          I2NSF Client                     | App Gateway
               +-------------------------------------------+
                                       |    Client Facing Interface
               +-------------------------------------------+
         Vendor |             Application Support           |
      Facing<->|-------------------------------------------| Security
      Interface| IKE Credential,PAD and SPD entries Distr. | Controller
               +-------------------------------------------+
                                       |       NSF Facing Interface
               +-------------------------------------------+
               |                 I2NSF Agent               |
               |-------------------------------------------| Network
               |   IKE     |      IPsec(SPD,PAD)           | Security
               |-------------------------------------------| Function
               |         Data Protection and Forwarding    |
               +-------------------------------------------+
```

Figure 1: IKE case: IKE/IPsec in the NSF

5.1.1. Interface Requirements for IKE case

SDN-based IPsec flow protection services provide dynamic and flexible management of IPsec SAs in flow-based NSFs. In order to support this capability in IKE case, the following interface requirements need to be met:

- A YANG data model for IKEv2, SPD and PAD configuration data, and for IKE state data.

- In scenarios where multiple Security Controllers are implicated, SDN-based IPsec management services may require a mechanism to discover which Security Controller is managing a specific NSF. Moreover, an east-west interface [RFC7426] is required to exchange IPsec-related information. For example, if two gateways need to establish an IPsec SA and both are under the control of two different controllers, then both Security Controllers need to exchange information to properly configure their own NSFs. That is, the may need to agree on whether IKEv2 authentication will be based on raw public keys, pre-shared keys, etc. In case of using pre-shared keys they will have to agree in the PSK.

5.2. IKE-less case: IPsec (no IKEv2) in the NSF.

In this case, the NSF does not deploy IKEv2 and, therefore, the Security Controller has to perform the IKE security functions and

management of IPsec SAs by populating and managing the SPD and the SAD.

```
             +-----------------------------------------+
             |   IPsec Management  Application         | Client or
             |               I2NSF Client              | App Gateway
             +-----------------------------------------+
                                 |   Client Facing Interface
             +-----------------------------------------+
       Vendor|             Application Support         |
    Facing<->|-----------------------------------------| Security
    Interface|      SPD, SAD and PAD Entries Distr.    | Controller
             +-----------------------------------------+
                                 |   NSF Facing Interface
             +-----------------------------------------+
             |              I2NSF Agent                | Network
             |-----------------------------------------| Security
             |            IPsec (SPD,SAD)              | Function (NSF)
             |-----------------------------------------|
             |     Data Protection and Forwarding      |
             +-----------------------------------------+
```

Figure 2: IKE-less case: IPsec (no IKE) in the NSF

As shown in Figure 2, applications for flow protection run on the top of the Security Controller. When an administrator enforces flow-based protection policies through the Client Facing Interface, the Security Controller translates these requirements into SPD and SAD entries, which are installed in the NSF. PAD entries are not required since there is no IKEv2 in the NSF.

5.2.1. Interface Requirements for IKE-less case

In order to support the IKE-less case, the following requirements need to be met:

- A YANG data model for configuration data for SPD and SAD and for state data for SAD.

- In scenarios where multiple controllers are implicated, SDN-based IPsec management services may require a mechanism to discover which Security Controller is managing a specific NSF. Moreover, an east-west interface [RFC7426] is required to exchange IPsec-related information. NOTE: A possible east-west protocol for this IKE-less case could be IKEv2. However, this needs to be explore since the IKEv2 peers would be the Security Controllers.

Specifically, the IKE-less case assumes that the SDN controller has to perform some security functions that IKEv2 typically does, namely (non-exhaustive):

- IV generation.
- Prevent counter resets for the same key.
- Generation of pseudo-random cryptographic keys for the IPsec SAs.
- Rekey of the IPsec SAs based on notifications from the NSF (i.e. expire).
- Generation of the IPsec SAs when required based on notifications (i.e. sadb-acquire) from the NSF.
- NAT Traversal discovery and management.

Additionally to these functions, another set of tasks must be performed by the Security Controller (non-exhaustive list):

- IPsec SA's SPI random generation.
- Cryptographic algorithm/s selection.
- Usage of extended sequence numbers.
- Establishment of proper traffic selectors.

5.3. IKE case vs IKE-less case

In principle, IKE case is easier to deploy than IKE-less case because current gateways typically have an IKEv2/IPsec implementation. Moreover hosts can install easily an IKE implementation. As downside, the NSF needs more resources to hold IKEv2. Moreover, the IKEv2 implementation needs to implement an internal interface so that the IKE configuration sent by the Security Controller can be enforced in runtime.

Alternatively, IKE-less case allows lighter NSFs (no IKEv2 implementation), which benefits the deployment in constrained NSFs. Moreover, IKEv2 does not need to be performed in gateway-to-gateway and host-to-host scenarios under the same Security Controller (see Section 7.1). On the contrary, the overload of creating and managing IPsec SAs is shifted to the Security Controller since IKEv2 is not in the NSF. As a consequence, this may result in a more complex implementation in the controller side in comparison with IKE case. For example, the Security Controller have to deal with the latency

existing in the path between the Security Controller and the NSF in order to solve tasks such as, rekey or creation and installation of new IPsec SAs. However, this is not specific to our contribution but a general aspect in any SDN-based network. In summary, this overload may create some scalability and performance issues when the number of NSFs is high.

Nevertheless, literature around SDN-based network management using a centralized Security Controller is aware about scalability and performance issues and solutions have been already provided and discussed (e.g. hierarchical Security Controllers; having multiple replicated Security Controllers, dedicated high-speed management networks, etc). In the context of SDN-based IPsec management, one way to reduce the latency and alleviate some performance issues can be the installation of the IPsec policies and IPsec SAs at the same time (proactive mode, as described in Section 7.1) instead of waiting for notifications (e.g. a notification sadb-acquire when a new IPsec SA is required) to proceed with the IPsec SA installations (reactive mode). Another way to reduce the overhead and the potential scalability and performance issues in the Security Controller is to apply the IKE case described in this document, since the IPsec SAs are managed between NSFs without the involvement of the Security Controller at all, except by the initial IKE configuration provided by the Security Controller. Other solutions, such as Controller-IKE [I-D.carrel-ipsecme-controller-ike], have proposed that NSFs provide their DH public keys to the Security Controller, so that the Security Controller distributes all public keys to all peers. All peers can calculate a unique pairwise secret for each other peer and there is no inter-NSF messages. A rekey mechanism is further described in [I-D.carrel-ipsecme-controller-ike].

In terms of security, IKE case provides better security properties than IKE-less case, as we discuss in section Section 9. The main reason is that the NSFs are generating the session keys and not the Security Controller.

5.3.1. Rekeying process.

For IKE case, the rekeying process is carried out by IKEv2, following the information defined in the SPD and SAD. Therefore, connections will live unless something different is required by the administrator or the Security Controller detects something wrong.

Traditionally, during a rekey process of the IPSec SA using IKE, a bundle of inbound and outbound IPsec SAs is taken into account from the perspective of one of the NSFs. For example, if the inbound IPsec SA expires both the inbound and outbound IPsec SA are rekeyed at the same time in that NSF. However, when IKE is not used, we have

followed a different approach to avoid any packet loss during rekey: the Security Controller installs first the new inbound SAs in both NSFs and then, the outbound IPsec SAs.

In other words, for the IKE-less case, the Security Controller needs to take care of the rekeying process. When the IPsec SA is going to expire (e.g. IPsec SA soft lifetime), it has to create a new IPsec SA and remove the old one. This rekeying process starts when the Security Controller receives a sadb-expire notification or it decides so, based on lifetime state data obtained from the NSF.

To explain the rekeying process between two IPsec NSFs A and B, let assume that SPIa1 identifies the inbound IPsec SA in A, and SPIb1 the inbound IPsec SA in B. The rekeying process will take the following steps:

1. The Security Controller chooses two random values as SPI for the new inbound IPsec SAs: for example, SPIa2 for A and SPIb2 for B. These numbers MUST NOT be in conflict with any IPsec SA in A or B. Then, the Security Controller creates an inbound IPsec SA with SPIa2 in A and another inbound IPsec SA in B with SPIb2. It can send this information simultaneously to A and B.

2. Once the Security Controller receives confirmation from A and B, the controller knows that the inbound IPsec A are correctly installed. Then it proceeds to send in parallel to A and B, the outbound IPsec SAs: it sends the outbound IPsec SA to A with SPIb2 and the outbound IPsec SA to B with SPIa2. At this point the new IPsec SAs are ready.

3. Once the Security Controller receives confirmation from A and B that the outbound IPsec SAs have been installed, the Security Controller, in parallel, deletes the old IPsec SAs from A (inbound SPIa1 and outbound SPIb1) and B (outbound SPIa1 and inbound SPIb1).

If some of the operations in step 1 fail (e.g. the NSF A reports an error when the Security Controller is trying to install a new inbound IPsec SA) the Security Controller must perform rollback operations by removing any new inbound SA that had been successfully installed during step 1.

If step 1 is successful but some of the operations in step 2 fails (e.g. the NSF A reports an error when the Security Controller is trying to install the new outbound IPsec SA), the Security Controller must perform a rollback operation by deleting any new outbound SA that had been successfully installed during step 2 and by deleting the inbound SAs created in step 1.

If the steps 1 an 2 are successful and the step 3 fails the Security Controller will avoid any rollback of the operations carried out in step 1 and step 2 since new and valid IPsec SAs were created and are functional. The Security Controller may reattempt to remove the old inbound and outbound SAs in NSF A and NSF B several times until it receives a success or it gives up. In the last case, the old IPsec SAs will be removed when the hard lifetime is reached.

5.3.2. NSF state loss.

If one of the NSF restarts, it will lose the IPsec state (affected NSF). By default, the Security Controller can assume that all the state has been lost and therefore it will have to send IKEv2, SPD and PAD information to the NSF in the IKE case, and SPD and SAD information in IKE-less case.

In both cases, the Security Controller is aware of the affected NSF (e.g. the NETCONF/TCP connection is broken with the affected NSF, the Security Controller is receiving sadb-bad-spi notification from a particular NSF, etc.). Moreover, the Security Controller has a register about all the NSFs that have IPsec SAs with the affected NSF. Therefore, it knows the affected IPsec SAs.

In IKE case, the Security Controller will configure the affected NSF with the new IKEv2, SPD and PAD information. It has also to send new parameters (e.g. a new fresh PSK for authentication) to the NSFs which have IKEv2 SAs and IPsec SAs with the affected NSF. Finally, the Security Controller will instruct the affected NSF to start the IKEv2 negotiation with the new configuration.

In IKE-less case, if the Security Controller detects that a NSF has lost the IPsec SAs it will delete the old IPsec SAs on the non-failed nodes, established with the failed node (step 1). This prevents the non-failed nodes from leaking plaintext. If the affected node comes to live, the Security Controller will configure the new inbound IPsec SAs between the affected node and all the nodes it was talking to (step 2). After these inbound IPsec SAs have been established, the Security Controller can configure the outbound IPsec SAs in parallel (step 3).

Nevertheless other more optimized options can be considered (e.g. making the IKEv2 configuration permanent between reboots).

5.3.3. NAT Traversal

In the IKE case, IKEv2 already provides a mechanism to detect whether some of the peers or both are located behind a NAT. If there is a NAT network configured between two peers, it is required to activate

the usage of UDP or TCP/TLS encapsulation for ESP packets ([RFC3948], [RFC8229]). Note that the usage of IPsec transport mode when NAT is required MUST NOT be used in this specification.

On the contrary, the IKE-less case does not have any protocol in the NSFs to detect whether they are located behind a NAT or not. However, the SDN paradigm generally assumes the Security Controller has a view of the network under its control. This view is built either requesting information to the NSFs under its control, or because these NSFs inform the Security Controller. Based on this information, the Security Controller can guess if there is a NAT configured between two hosts, and apply the required policies to both NSFs besides activating the usage of UDP or TCP/TLS encapsulation of ESP packets ([RFC3948], [RFC8229]).

For example, the Security Controller could directly request the NSF for specific data such as networking configuration, NAT support, etc. Protocols such as NETCONF or SNMP can be used here. For example, RFC 7317 [RFC7317] provides a YANG data model for system management or [RFC8512] a data model for NAT management. The Security Controller can use this NETCONF module with a NSF to collect NAT information or even configure a NAT network. In any case, if this NETCONF module is not available in the NSF and the Security Controller does not have a mechanism to know whether a host is behind a NAT or not, then the IKE case should be the right choice and not the IKE-less case.

5.3.4. NSF Discovery

The assumption in this document is that, for both cases, before a NSF can operate in this system, it MUST be registered in the Security Controller. In this way, when the NSF comes to live and establishes a connection to the Security Controller, it knows that the NSF is valid for joining the system.

Either during this registration process or when the NSF connects with the Security Controller, the Security Controller MUST discover certain capabilities of this NSF, such as what is the cryptographic suite supported, authentication method, the support of the IKE case or the IKE-less case, etc. This discovery process is out of the scope of this document.

6. YANG configuration data models

In order to support the IKE and IKE-less cases we have modeled the different parameters and values that must be configured to manage IPsec SAs. Specifically, IKE requires modeling IKEv2, SPD and PAD, while IKE-less case requires configuration models for the SPD and SAD. We have defined three models: ietf-ipsec-common (Appendix A),

ietf-ipsec-ike (Appendix B, IKE case), ietf-ipsec-ikeless (Appendix C, IKE-less case). Since the model ietf-ipsec-common has only typedef and groupings common to the other modules, we only show a simplified view of the ietf-ipsec-ike and ietf-ipsec-ikeless models.

6.1. IKE case model

The model related to IKEv2 has been extracted from reading IKEv2 standard in [RFC7296], and observing some open source implementations, such as Strongswan [strongswan] or Libreswan [libreswan].

The definition of the PAD model has been extracted from the specification in section 4.4.3 in [RFC4301] (NOTE: We have observed that many implementations integrate PAD configuration as part of the IKEv2 configuration).

```
module: ietf-ipsec-ike
+--rw ipsec-ike
  +--rw pad
  |  +--rw pad-entry* [name]
  |     +--rw name                          string
  |     +--rw (identity)
  |     |  +--:(ipv4-address)
  |     |  |  +--rw ipv4-address?           inet:ipv4-address
  |     |  +--:(ipv6-address)
  |     |  |  +--rw ipv6-address?           inet:ipv6-address
  |     |  +--:(fqdn-string)
  |     |  |  +--rw fqdn-string?            inet:domain-name
  |     |  +--:(rfc822-address-string)
  |     |  |  +--rw rfc822-address-string?   string
  |     |  +--:(dnx509)
  |     |  |  +--rw dnx509?                 string
  |     |  +--:(gnx509)
  |     |  |  +--rw gnx509?                 string
  |     |  +--:(id-key)
  |     |  |  +--rw id-key?                 string
  |     |  +--:(id-null)
  |     |     +--rw id-null?                empty
  |     +--rw auth-protocol?                auth-protocol-type
  |     +--rw peer-authentication
  |        +--rw auth-method?               auth-method-type
  |        +--rw eap-method
  |        |  +--rw eap-type                uint8
  |        +--rw pre-shared
```

```
     |        |  +--rw secret?                  yang:hex-string
     |        +--rw digital-signature
     |           +--rw ds-algorithm?            uint8
     |           +--rw (public-key)
     |           |  +--:(raw-public-key)
     |           |  |  +--rw raw-public-key?   binary
     |           |  +--:(cert-data)
     |           |     +--rw cert-data?        ct:x509
     |           +--rw private-key?            binary
     |           +--rw ca-data*                ct:x509
     |           +--rw crl-data?               ct:crl
     |           +--rw crl-uri?                inet:uri
     |           +--rw oscp-uri?               inet:uri
     +--rw conn-entry* [name]
     |  +--rw name                                 string
     |  +--rw autostartup?                         autostartup-type
     |  +--rw initial-contact?                     boolean
     |  +--rw version?                             auth-protocol-type
     |  +--rw fragmentation?                       boolean
     |  +--rw ike-sa-lifetime-soft
     |  |  +--rw rekey-time?    uint32
     |  |  +--rw reauth-time?   uint32
     |  +--rw ike-sa-lifetime-hard
     |  |  +--rw over-time?   uint32
     |  +--rw authalg*  ic:integrity-algorithm-type
     |  +--rw encalg*   ic:encryption-algorithm-type
     |  +--rw dh-group?                            pfs-group
     |  +--rw half-open-ike-sa-timer?              uint32
     |  +--rw half-open-ike-sa-cookie-threshold?   uint32
     |  +--rw local
     |  |  +--rw local-pad-entry-name?   string
     |  +--rw remote
     |  |  +--rw remote-pad-entry-name?   string
     |  +--rw encapsulation-type
     |  |  +--rw espencap?   esp-encap
     |  |  +--rw sport?      inet:port-number
     |  |  +--rw dport?      inet:port-number
     |  |  +--rw oaddr*      inet:ip-address
     |  +--rw spd
     |  |  +--rw spd-entry* [name]
     |  |    +--rw name                   string
     |  |    +--rw ipsec-policy-config
     |  |      +--rw anti-replay-window?   uint64
     |  |      +--rw traffic-selector
     |  |      |  +--rw local-subnet      inet:ip-prefix
     |  |      |  +--rw remote-subnet     inet:ip-prefix
     |  |      |  +--rw inner-protocol?   ipsec-inner-protocol
     |  |      |  +--rw local-ports* [start end]
```

```
     |  |      |  |  +--rw start          inet:port-number
     |  |      |  |  +--rw end            inet:port-number
     |  |      |  +--rw remote-ports* [start end]
     |  |      |     +--rw start          inet:port-number
     |  |      |     +--rw end            inet:port-number
     |  |      +--rw processing-info
     |  |      |  +--rw action?         ipsec-spd-action
     |  |      |  +--rw ipsec-sa-cfg
     |  |      |    +--rw pfp-flag?              boolean
     |  |      |    +--rw ext-seq-num?           boolean
     |  |      |    +--rw seq-overflow?          boolean
     |  |      |    +--rw stateful-frag-check?   boolean
     |  |      |    +--rw mode?                  ipsec-mode
     |  |      |    +--rw protocol-parameters? ipsec-protocol-parameters
     |  |      |    +--rw esp-algorithms
     |  |      |    |  +--rw integrity*  integrity-algorithm-type
     |  |      |    |  +--rw encryption* encryption-algorithm-type
     |  |      |    |  +--rw tfc-pad?      boolean
     |  |      |    +--rw tunnel
     |  |      |       +--rw local           inet:ip-address
     |  |      |       +--rw remote          inet:ip-address
     |  |      |       +--rw df-bit?         enumeration
     |  |      |       +--rw bypass-dscp?    boolean
     |  |      |       +--rw dscp-mapping?   yang:hex-string
     |  |      |       +--rw ecn?            boolean
     |  |      +--rw spd-mark
     |  |         +--rw mark?   uint32
     |  |         +--rw mask?   yang:hex-string
     |  +--rw child-sa-info
     |  |  +--rw pfs-groups*               pfs-group
     |  |  +--rw child-sa-lifetime-soft
     |  |  |  +--rw time?      uint32
     |  |  |  +--rw bytes?     uint32
     |  |  |  +--rw packets?   uint32
     |  |  |  +--rw idle?      uint32
     |  |  |  +--rw action?    ic:lifetime-action
     |  |  +--rw child-sa-lifetime-hard
     |  |     +--rw time?      uint32
     |  |     +--rw bytes?     uint32
     |  |     +--rw packets?   uint32
     |  |     +--rw idle?      uint32
     |  +--ro state
     |     +--ro initiator?             boolean
     |     +--ro initiator-ikesa-spi?   ike-spi
     |     +--ro responder-ikesa-spi?   ike-spi
     |     +--ro nat-local?             boolean
     |     +--ro nat-remote?            boolean
     |     +--ro encapsulation-type
```

```
     |     |  +--ro espencap?   esp-encap
     |     |  +--ro sport?      inet:port-number
     |     |  +--ro dport?      inet:port-number
     |     |  +--ro oaddr*      inet:ip-address
     |     +--ro established?           uint64
     |     +--ro current-rekey-time?    uint64
     |     +--ro current-reauth-time?   uint64
     +--ro number-ike-sas
        +--ro total?                uint64
        +--ro half-open?            uint64
        +--ro half-open-cookies?    uint64
```

Appendix D shows an example of IKE case configuration for a NSF, in tunnel mode (gateway-to-gateway), with NSFs authentication based on X.509 certificates.

6.2. IKE-less case model

For this case, the definition of the SPD model has been mainly extracted from the specification in section 4.4.1 and Appendix D in [RFC4301], though with some simplications. For example, each IPsec policy (spd-entry) contains one traffic selector, instead a list of them. The reason is that we have observed real kernel implementations only admit a traffic selector per IPsec policy.

The definition of the SAD model has been extracted from the specification in section 4.4.2 in [RFC4301]. Note that this model not only allows to associate an IPsec SA with its corresponding policy through the specific traffic selector but also an identifier (reqid).

The notifications model has been defined using as reference the PF_KEYv2 standard in [RFC2367].

```
   module: ietf-ipsec-ikeless
   +--rw ipsec-ikeless
    +--rw spd
    |  +--rw spd-entry* [name]
    |     +--rw name                     string
    |     +--rw direction?               ic:ipsec-traffic-direction
    |     +--rw reqid?                   uint64
    |     +--rw ipsec-policy-config
    |        +--rw anti-replay-window?   uint64
    |        +--rw traffic-selector
    |        |  +--rw local-subnet      inet:ip-prefix
    |        |  +--rw remote-subnet     inet:ip-prefix
```

```
    |        |  +--rw inner-protocol?   ipsec-inner-protocol
    |        |  +--rw local-ports* [start end]
    |        |  |  +--rw start          inet:port-number
    |        |  |  +--rw end            inet:port-number
    |        |  +--rw remote-ports* [start end]
    |        |     +--rw start          inet:port-number
    |        |     +--rw end            inet:port-number
    |        +--rw processing-info
    |        |  +--rw action?         ipsec-spd-action
    |        |  +--rw ipsec-sa-cfg
    |        |     +--rw pfp-flag?              boolean
    |        |     +--rw ext-seq-num?           boolean
    |        |     +--rw seq-overflow?          boolean
    |        |     +--rw stateful-frag-check?   boolean
    |        |     +--rw mode?                  ipsec-mode
    |        |     +--rw protocol-parameters?
    |        |     +--rw esp-algorithms
    |        |     |  +--rw integrity*    integrity-algorithm-type
    |        |     |  +--rw encryption*  encryption-algorithm-type
    |        |     |  +--rw tfc-pad?      boolean
    |        |     +--rw tunnel
    |        |        +--rw local           inet:ip-address
    |        |        +--rw remote          inet:ip-address
    |        |        +--rw df-bit?         enumeration
    |        |        +--rw bypass-dscp?    boolean
    |        |        +--rw dscp-mapping?   yang:hex-string
    |        |        +--rw ecn?            boolean
    |        +--rw spd-mark
    |           +--rw mark?   uint32
    |           +--rw mask?   yang:hex-string
    +--rw sad
     +--rw sad-entry* [name]
       +--rw name               string
       +--rw reqid?             uint64
       +--rw ipsec-sa-config
       |  +--rw spi                    uint32
       |  +--rw ext-seq-num?           boolean
       |  +--rw seq-number-counter?    uint64
       |  +--rw seq-overflow?          boolean
       |  +--rw anti-replay-window?    uint32
       |  +--rw traffic-selector
       |  |  +--rw local-subnet       inet:ip-prefix
       |  |  +--rw remote-subnet      inet:ip-prefix
       |  |  +--rw inner-protocol?    ipsec-inner-protocol
       |  |  +--rw local-ports*       [start end]
       |  |  |  +--rw start           inet:port-number
       |  |  |  +--rw end             inet:port-number
       |  |  +--rw remote-ports* [start end]
```

```
       |  |     +--rw start           inet:port-number
       |  |     +--rw end             inet:port-number
       |  +--rw protocol-parameters?  ic:ipsec-protocol-parameters
       |  +--rw mode?                 ic:ipsec-mode
       |  +--rw esp-sa
       |  |  +--rw encryption
       |  |  |  +--rw encryption-algorithm? ic:encryption-algorithm-type
       |  |  |  +--rw key?                    yang:hex-string
       |  |  |  +--rw iv?                     yang:hex-string
       |  |  +--rw integrity
       |  |     +--rw integrity-algorithm? ic:integrity-algorithm-type
       |  |     +--rw key?                   yang:hex-string
       |  +--rw sa-lifetime-hard
       |  |  +--rw time?      uint32
       |  |  +--rw bytes?     uint32
       |  |  +--rw packets?   uint32
       |  |  +--rw idle?      uint32
       |  +--rw sa-lifetime-soft
       |  |  +--rw time?      uint32
       |  |  +--rw bytes?     uint32
       |  |  +--rw packets?   uint32
       |  |  +--rw idle?      uint32
       |  |  +--rw action?    ic:lifetime-action
       |  +--rw tunnel
       |  |  +--rw local           inet:ip-address
       |  |  +--rw remote          inet:ip-address
       |  |  +--rw df-bit?         enumeration
       |  |  +--rw bypass-dscp?    boolean
       |  |  +--rw dscp-mapping?   yang:hex-string
       |  |  +--rw ecn?            boolean
       |  +--rw encapsulation-type
       |     +--rw espencap?   esp-encap
       |     +--rw sport?      inet:port-number
       |     +--rw dport?      inet:port-number
       |     +--rw oaddr*      inet:ip-address
       +--ro ipsec-sa-state
          +--ro sa-lifetime-current
          |  +--ro time?      uint32
          |  +--ro bytes?     uint32
          |  +--ro packets?   uint32
          |  +--ro idle?      uint32
          +--ro replay-stats
             +--ro replay-window?        uint64
             +--ro packet-dropped?       uint64
             +--ro failed?               uint32
             +--ro seq-number-counter?   uint64

    notifications:
```

```
     +---n sadb-acquire
     |  +--ro ipsec-policy-name          string
     |  +--ro traffic-selector
     |     +--ro local-subnet            inet:ip-prefix
     |     +--ro remote-subnet           inet:ip-prefix
     |     +--ro inner-protocol?         ipsec-inner-protocol
     |     +--ro local-ports* [start end]
     |     |  +--ro start                inet:port-number
     |     |  +--ro end                  inet:port-number
     |     +--ro remote-ports* [start end]
     |        +--ro start                inet:port-number
     |        +--ro end                  inet:port-number
     +---n sadb-expire
     |  +--ro ipsec-sa-name           string
     |  +--ro soft-lifetime-expire?   boolean
     |  +--ro lifetime-current
     |     +--ro time?                uint32
     |     +--ro bytes?               uint32
     |     +--ro packets?             uint32
     |     +--ro idle?                uint32
     +---n sadb-seq-overflow
     |  +--ro ipsec-sa-name           string
     +---n sadb-bad-spi
        +--ro spi                     uint32
```

Appendix E shows an example of IKE-less case configuration for a NSF, in transport mode (host-to-host), with NSFs authentication based on shared secrets. For the IKE-less case, Appendix F shows examples of IPsec SA expire, acquire, sequence number overflow and bad SPI notifications.

7. Use cases examples

This section explains how different traditional configurations, that is, host-to-host and gateway-to-gateway, are deployed using this SDN-based IPsec management service. In turn, these configurations will be typical in modern networks where, for example, virtualization will be key.

7.1. Host-to-host or gateway-to-gateway under the same Security Controller

```
                      +----------------------------------------+
                      |           Security Controller          |
                      |                                        |
                   (1)|   +--------------+ (2)+--------------+ |
      Flow-based  ------> |Translate into|--->| South. Prot. | |
      Security. Pol.  |   |IPsec Policies|    |              | |
                      |   +--------------+    +--------------+ |
                      |                          |     |       |
                      |                          |     |       |
                      +--------------------------|-----|-------+
                                                 |     |
                                                 | (3) |
                       |-------------------------+     +---|
                       V                                   V
           +----------------------+         +----------------------+
           |    NSF A             |<=======>|   NSF B              |
           |IKEv2/IPsec(SPD/PAD)  |         |IKEv2/IPsec(SPD/PAD)  |
           +----------------------+  (4)    +----------------------+
```

Figure 3: Host-to-host / gateway-to-gateway single Security Controller for the IKE case.

Figure 3 describes the IKE case:

1. The administrator defines general flow-based security policies. The Security Controller looks for the NSFs involved (NSF A and NSF B).

2. The Security Controller generates IKEv2 credentials for them and translates the policies into SPD and PAD entries.

3. The Security Controller inserts an IKEv2 configuration that includes the SPD and PAD entries in both NSF A and NSF B. If some of operations with NSF A and NSF B fail the Security Controller will stop the process and perform a rollback operation by deleting any IKEv2, SPD and PAD configuration that had been successfully installed in NSF A or B.

4. If the previous step is successful, the flow is protected by means of the IPsec SA established with IKEv2.

```
                     +----------------------------------------+
                     |    (1)     Security Controller         |
         Flow-based  |                                        |
         Security -----------|                                |
         Policy      |       V                                |
                     |  +---------------+ (2)+-------------+  |
                     |  |Translate into |--->| South. Prot.|  |
                     |  |IPsec policies |    |             |  |
                     |  +---------------+    +-------------+  |
                     |                         |     |        |
                     |                         |     |        |
                     +-------------------------| --- |--------+
                                               |     |
                                               | (3) |
                        |----------------------+     +--|
                        V                               V
               +------------------+      +------------------+
               |    NSF A         |<=====>|   NSF B          |
               |IPsec(SPD/SAD)    |   4)  |IPsec(SPD/SAD)    |
               +------------------+      +------------------+
```

Figure 4: Host-to-host / gateway-to-gateway single Security Controller for IKE-less case.

In the IKE-less case, flow-based security policies defined by the administrator are translated into IPsec SPD entries and inserted into the corresponding NSFs. Besides, fresh SAD entries will be also generated by the Security Controller and enforced in the NSFs. In this case, the Security Controller does not run any IKEv2 implementation (neither the NSFs), and it provides the cryptographic material for the IPsec SAs. These keys will be also distributed securely through the southbound interface. Note that this is possible because both NSFs are managed by the same Security Controller.

Figure 4 describes the IKE-less case, when a data packet needs to be protected in the path between the NSF A and NSF B:

1. The administrator establishes the flow-based security policies, and the Security Controller looks for the involved NSFs.

2. The Security Controller translates the flow-based security policies into IPsec SPD and SAD entries.

3. The Security Controller inserts these entries in both NSF A and NSF B IPsec databases (SPD and SAD). The following text describes how this happens between two NSFs A and B:

- The Security Controller chooses two random values as SPIs: for example, SPIa1 for NSF A and SPIb1 for NSF B. These numbers MUST NOT be in conflict with any IPsec SA in NSF A or NSF B. It also generates fresh cryptographic material for the new inbound/outbound IPsec SAs and their parameters and send simultaneously the new inbound IPsec SA with SPIa1 and new outbound IPsec SAs with SPIb1 to NSF A; and the new inbound IPsec SA with SPIb1 and new outbound IPsec SAs with SPIa1 to B, together with the corresponding IPsec policies.

- Once the Security Controller receives confirmation from NSF A and NSF B, the controller knows that the IPsec SAs are correctly installed and ready.

If some of the operations described above fails (e.g. the NSF A reports an error when the Security Controller is trying to install the SPD entry, the new inbound and outbound IPsec SAs) the Security Controller must perform rollback operations by deleting any new inbound or outbound SA and SPD entry that had been successfully installed in any of the NSFs (e.g NSF B) and stop the process (NOTE: the Security Controller may retry several times before giving up). Other alternative to this operation is: the Security Controller sends first the IPsec policies and new inbound IPsec SAs to A and B and once it obtains a successful confirmation of these operations from NSF A and NSF B, it proceeds with installing to the new outbound IPsec SAs. However, this may increase the latency to complete the process. As an advantage, no traffic is sent over the network until the IPsec SAs are completely operative. In any case other alternatives may be possible. Finally, it is worth mentioning that the Security Controller associates a lifetime to the new IPsec SAs. When this lifetime expires, the NSF will send a sadb-expire notification to the Security Controller in order to start the rekeying process.

4. The flow is protected with the IPsec SA established by the Security Controller.

Instead of installing IPsec policies in the SPD and IPsec SAs in the SAD in step 3 (proactive mode), it is also possible that the Security Controller only installs the SPD entries in step 3 (reactive mode). In such a case, when a data packet requires to be protected with IPsec, the NSF that saw first the data packet will send a sadb-acquire notification that informs the Security Controller that needs SAD entries with the IPsec SAs to process the data packet. In such as reactive mode, since IPsec policies are already installed in the SPD, the Security Controller installs first the new IPsec SAs in NSF A and B with the operations described in step 3 but without sending any IPsec policies. Again, if some of the operations installing the

new inbound/outbound IPsec SAs fail, the Security Controller stops the process and performs a rollback operation by deleting any new inbound/outbound SAs that had been successfully installed.

Both NSFs could be two hosts that exchange traffic and require to establish an end-to-end security association to protect their communications (host-to-host) or two gateways (gateway-to-gateway), for example, within an enterprise that needs to protect the traffic between the networks of two branch offices.

Applicability of these configurations appear in current and new networking scenarios. For example, SD-WAN technologies are providing dynamic and on-demand VPN connections between branch offices, or between branches and SaaS cloud services. Beside, IaaS services providing virtualization environments are deployments solutions based on IPsec to provide secure channels between virtual instances (host-to-host) and providing VPN solutions for virtualized networks (gateway-to-gateway).

In general (for IKE and IKE-less cases), this system has various advantages:

1. It allows to create IPsec SAs among two NSFs, based only on the application of general Flow-based Security Policies at the application layer. Thus, administrators can manage all security associations in a centralized point with an abstracted view of the network.

2. Any NSF deployed in the system does not need manual configuration, therefore allowing its deployment in an automated manner.

7.2. Host-to-host or gateway-to-gateway under different Security Controllers

It is also possible that two NSFs (i.e. NSF A and NSF B) are under the control of two different Security Controllers. This may happen, for example, when two organizations, namely Enterprise A and Enterprise B, have their headquarters interconnected through a WAN connection and they both have deployed a SDN-based architecture to provide connectivity to all their clients.

```
                  +-------------+           +-------------+
                  |             |           |             |
        Flow-based|   Security  |<=========>|   Security <--Flow-based
        Sec. Pol.--> Controller |  (3)      |  Controller | Sec. Pol.
              (1) |      A      |           |      B      |   (2)
                  +-------------+           +-------------+
                          |                        |
                          | (4)                (4) |
                          V                        V
              +--------------------+          +--------------------+
              |    NSF A            |<=======>|   NSF B            |
              |IKEv2/IPsec(SPD/PAD)|          |IKEv2/IPsec(SPD/PAD)|
              +--------------------+  (5)     +--------------------+
```

Figure 5: Different Security Controllers in the IKE case.

Figure 5 describes IKE case when two Security Controllers are involved in the process.

1. The A's administrator establishes general Flow-based Security Policies in Security Controller A.

2. The B's administrator establishes general Flow-based Security Policies in Security Controller B.

3. The Security Controller A realizes that protection is required between the NSF A and NSF B, but the NSF B is under the control of another Security Controller (Security Controller B), so it starts negotiations with the other controller to agree on the IPsec SPD policies and IKEv2 credentials for their respective NSFs. NOTE: This may require extensions in the East/West interface.

4. Then, both Security Controllers enforce the IKEv2 credentials, related parameters and the SPD and PAD entries in their respective NSFs.

5. The flow is protected with the IPsec SAs established with IKEv2 between both NSFs.

```
                   +--------------+             +--------------+
                   |              |             |              |
         Flow-based. --->        |             |         <---Flow-based
            Prot.  |   Security   |<===========>|   Security   |Sec.
            Pol.(1)|  Controller  |     (3)     |  Controller  |Pol. (2)
                   |       A      |             |       B      |
                   +--------------+             +--------------+
                           |                               |
                           | (4)                       (4) |
                           V                               V
                  +--------------+      (5)       +--------------+
                  |    NSF A     |<==============>|    NSF B     |
                  |IPsec(SPD/SAD)|                |IPsec(SPD/SAD)|
                  +--------------+                +--------------+
```

Figure 6: Different Security Controllers in the IKE-less case.

Figure 6 describes IKE-less case when two Security Controllers are involved in the process.

1. The A's administrator establishes general Flow Protection Policies in Security Controller A.

2. The B's administrator establishes general Flow Protection Policies in Security Controller B.

3. The Security Controller A realizes that the flow between NSF B and NSF B MUST be protected. Nevertheless, it notices that NSF B is under the control of another Security Controller B, so it starts negotiations with the other controller to agree on the IPsec SPD and SAD entries that define the IPsec SAs. NOTE: It would worth evaluating IKEv2 as the protocol for the East/West interface in this case.

4. Once the Security Controllers have agreed on the key material and the details of the IPsec SAs, they both enforce this information into their respective NSFs.

5. The flow is protected with the IPsec SAs established by both Security Controllers in their respective NSFs.

8. IANA Considerations

This document registers three URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-ipsec-common
Registrant Contact: The I2NSF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ipsec-ike
Registrant Contact: The I2NSF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless
Registrant Contact: The I2NSF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

This document registers three YANG modules in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

Name:	ietf-ipsec-common
Namespace:	urn:ietf:params:xml:ns:yang:ietf-ipsec-common
Prefix:	ic
Reference:	RFC XXXX

Name:	ietf-ipsec-ike
Namespace:	urn:ietf:params:xml:ns:yang:ietf-ipsec-ike
Prefix:	ike
Reference:	RFC XXXX

Name:	ietf-ipsec-ikeless
Namespace:	urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless
Prefix:	ikeless
Reference:	RFC XXXX

9. Security Considerations

First of all, this document shares all the security issues of SDN that are specified in the "Security Considerations" section of [ITU-T.Y.3300] and [RFC8192].

On the one hand, it is important to note that there MUST exit a security association between the Security Controller and the NSFs to protect of the critical information (cryptographic keys, configuration parameter, etc...) exchanged between these entities. For example, when NETCONF is used as southbound protocol between the Security Controller and the NSFs, it is defined that TLS or SSH security association MUST be established between both entities.

On the other hand, if encryption is mandatory for all traffic of a NSF, its default policy MUST be to drop (DISCARD) packets to prevent cleartext packet leaks. This default policy MUST be in the startup

configuration datastore in the NSF before the NSF contacts with the Security Controller. Moreover, the startup configuration datastore MUST be pre-configured with the required ALLOW policies that allow to communicate the NSF with the Security Controller once the NSF is deployed. This pre-configuration step is not carried out by the Security Controller but by some other entity before the NSF deployment. In this manner, when the NSF starts/reboots, it will always apply first the configuration in the startup configuration before contacting the Security Controller.

Finally, we have divided this section in two parts in order to analyze different security considerations for both cases: NSF with IKEv2 (IKE case) and NSF without IKEv2 (IKE-less case). In general, the Security Controller, as typically in the SDN paradigm, is a target for different type of attacks. Thus, the Security Controller is a key entity in the infrastructure and MUST be protected accordingly. In particular, the Security Controller will handle cryptographic material so that the attacker may try to access this information. Although we can assume this attack will not likely to happen due to the assumed security measurements to protect the Security Controller, it deserves some analysis in the hypothetical case the attack occurs. The impact is different depending on the IKE case or IKE-less case.

9.1. IKE case

In IKE case, the Security Controller sends IKE credentials (PSK, public/private keys, certificates, etc.) to the NSFs using the security association between Security Controller and NSFs. The general recommendation is that the Security Controller MUST NOT store the IKE credentials after distributing them. Moreover, the NSFs MUST NOT allow the reading of these values once they have been applied by the Security Controller (i.e. write only operations). One option is to return always the same value (i.e. all 0s) if a read operation is carried out. If the attacker has access to the Security Controller during the period of time that key material is generated, it might have access to the key material. Since these values are used during NSF authentication in IKEv2, it may impersonate the affected NSFs. Several recommendations are important. If PSK authentication is used in IKEv2, the Security Controller MUST remove the PSK immediately after generating and distributing it. Moreover, the PSK MUST have a proper length (e.g. minimum 128 bit length) and strength. When public/private keys are used, the Security Controller MAY generate both public key and private key. In such a case, the Security Controller MUST remove the associated private key immediately after distributing them to the NSFs. Alternatively, the NSF could generate the private key and export only the public key to the Security Controller. If certificates are used, the NSF MAY generate the

private key and exports the public key for certification to the Security Controller. How the NSF generates these cryptographic material (public key/private keys) and export the public key, or it is instructed to do so, it is out of the scope of this document.

9.2. IKE-less case

In the IKE-less case, the Security Controller sends the IPsec SA information to the NSF's SAD that includes the private session keys required for integrity and encryption. The general recommendation is that it MUST NOT store the keys after distributing them. Moreover, the NSFs receiving private key material MUST NOT allow the reading of these values by any other entity (including the Security Controller itself) once they have been applied (i.e. write only operations) into the NSFs. Nevertheless, if the attacker has access to the Security Controller during the period of time that key material is generated, it may obtain these values. In other words, the attacker might be able to observe the IPsec traffic and decrypt, or even modify and re-encrypt the traffic between peers.

9.3. YANG modules

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in these YANG modules that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

The YANG modules describe configuration data for the IKE case (ietf-ipsec-ike) and IKE-less case (ietf-ipsec-ikeless). There is a common module (ietf-ipsec-common) used in both cases.

For the IKE case (ietf-ipsec-ike):

/ipsec-ike: The entire container in this module is sensitive to write operations. An attacker may add/modify the credentials to be used for the authentication (e.g. to impersonate a NSF), the trust root (e.g. changing the trusted CA certificates), the cryptographic algorithms (allowing a downgrading attack), the IPsec policies (e.g. by allowing leaking of data traffic by changing to a allow policy), and in general changing the IKE SA conditions and credentials between any NSF.

For the IKE-less case (ietf-ipsec-ikeless):

/ipsec-ikeless: The entire container in this module is sensitive to write operations. An attacker may add/modify/delete any IPsec policies (e.g. by allowing leaking of data traffic by changing to a allow policy) in the /ipsec-ikeless/spd container, and add/modify/delete any IPsec SAs between two NSF by means of /ipsec-ikeless/sad container and, in general changing any IPsec SAs and IPsec policies between any NSF.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

For the IKE case (ietf-ipsec-ike):

/ipsec-ike/pad: This container includes sensitive information to read operations. This information should never be returned to a client. For example, cryptographic material configured in the NSFs: peer-authentication/pre-shared/secret and peer-authentication/digital-signature/private-key are already protected by the NACM extension "default-deny-all" in this document.

For the IKE-less case (ietf-ipsec-ikeless):

/ipsec-ikeless/sad/ipsec-sa-config/esp-sa: This container includes symmetric keys for the IPsec SAs. For example, encryption/key contains a ESP encryption key value and encryption/iv contains a initialization vector value. Similarly, integrity/key has ESP integrity key value. Those values must not be read by anyone and are protected by the NACM extension "default-deny-all" in this document.

10. Acknowledgements

Authors want to thank Paul Wouters, Valery Smyslov, Sowmini Varadhan, David Carrel, Yoav Nir, Tero Kivinen, Martin Bjorklund, Graham Bartlett, Sandeep Kampati, Linda Dunbar, Carlos J. Bernardos, Alejandro Perez-Mendez, Alejandro Abad-Carrascosa, Ignacio Martinez and Ruben Ricart for their valuable comments.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

11.2. Informative References

[I-D.carrel-ipsecme-controller-ike]
Carrel, D. and B. Weiss, "IPsec Key Exchange using a Controller", draft-carrel-ipsecme-controller-ike-01 (work in progress), March 2019.

[I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-08 (work in progress), July 2019.

[I-D.tran-ipsecme-yang]
Tran, K., Wang, H., Nagaraj, V., and X. Chen, "Yang Data Model for Internet Protocol Security (IPsec)", draft-tran-ipsecme-yang-01 (work in progress), June 2015.

[ITU-T.X.1252]
"Baseline Identity Management Terms and Definitions", April 2010.

[ITU-T.X.800]
"Security Architecture for Open Systems Interconnection for CCITT Applications", March 1991.

[ITU-T.Y.3300]
"Recommendation ITU-T Y.3300", June 2014.

[libreswan]
The Libreswan Project, "Libreswan VPN software", August 2019.

[netconf-vpn]
Stefan Wallin, "Tutorial: NETCONF and YANG", January 2014.

[ONF-OpenFlow]
ONF, "OpenFlow Switch Specification (Version 1.4.0)", October 2013.

[ONF-SDN-Architecture]
"SDN Architecture", June 2014.

[RFC2367] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API, Version 2", RFC 2367, DOI 10.17487/RFC2367, July 1998, <https://www.rfc-editor.org/info/rfc2367>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <https://www.rfc-editor.org/info/rfc3948>.

[RFC6071] Frankel, S. and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", RFC 6071, DOI 10.17487/RFC6071, February 2011, <https://www.rfc-editor.org/info/rfc6071>.

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <https://www.rfc-editor.org/info/rfc7149>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <https://www.rfc-editor.org/info/rfc7426>.

[RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, DOI 10.17487/RFC8192, July 2017, <https://www.rfc-editor.org/info/rfc8192>.

[RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <https://www.rfc-editor.org/info/rfc8229>.

[RFC8512] Boucadair, M., Ed., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", RFC 8512, DOI 10.17487/RFC8512, January 2019, <https://www.rfc-editor.org/info/rfc8512>.

[strongswan] CESNET, "StrongSwan: the OpenSource IPsec-based VPN Solution", August 2019.

Appendix A. Appendix A: Common YANG model for IKE and IKE-less cases

```
       <CODE BEGINS> file "ietf-ipsec-common@2019-08-05.yang"

       module ietf-ipsec-common {
           yang-version 1.1;
           namespace "urn:ietf:params:xml:ns:yang:ietf-ipsec-common";
           prefix "ipsec-common";

           import ietf-inet-types { prefix inet; }
           import ietf-yang-types { prefix yang; }

           organization "IETF I2NSF Working Group";

           contact
           "WG Web:  <https://datatracker.ietf.org/wg/i2nsf/about/>
            WG List: <mailto:i2nsf@ietf.org>

           Author: Rafael Marin-Lopez
                   <mailto:rafa@um.es>

           Author: Gabriel Lopez-Millan
                   <mailto:gabilm@um.es>

           Author: Fernando Pereniguez-Garcia
                   <mailto:fernando.pereniguez@cud.upct.es>
           ";

           description
               "Common Data model for the IKE and IKE-less cases
                defined by the SDN-based IPsec flow protection service.

               Copyright (c) 2019 IETF Trust and the persons
               identified as authors of the code.  All rights reserved.
               Redistribution and use in source and binary forms, with
               or without modification, is permitted pursuant to, and
               subject to the license terms contained in, the
               Simplified BSD License set forth in Section 4.c of the
               IETF Trust's Legal Provisions Relating to IETF Documents
               (https://trustee.ietf.org/license-info).

               This version of this YANG module is part of RFC XXXX;;
               see the RFC itself for full legal notices.

               The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
               'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
```

```
               'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this
               document are to be interpreted as described in BCP 14
               (RFC 2119) (RFC 8174) when, and only when, they appear
               in all capitals, as shown here.";

           revision "2019-08-05" {
               description "Revision 06";
               reference "RFC XXXX: YANG Groupings and typedef
                          for IKE and IKE-less case";
           }

           typedef encryption-algorithm-type {
               type uint16;
               description
                   "The encryption algorithm is specified with a 16-bit
                   number extracted from IANA Registry. The acceptable
                   values MUST follow the requirement levels for
                   encryption algorithms for ESP and IKEv2.";
               reference
                    "IANA Registry- Transform Type 1 - Encryption
                    Algorithm Transform IDs. RFC 8221 - Cryptographic
                    Algorithm Implementation Requirements and Usage
                    Guidance for Encapsulating Security Payload (ESP)
                    and Authentication Header (AH) and RFC 8247 -
                    Algorithm Implementation Requirements and Usage
                    Guidance for the Internet Key Exchange Protocol
                    Version 2 (IKEv2).";
           }

           typedef integrity-algorithm-type {
               type uint16;
               description
                   "The integrity algorithm is specified with a 16-bit
                   number extracted from IANA Registry.
                   The acceptable values MUST follow the requirement
                   levels for encryption algorithms for ESP and IKEv2.";
               reference
                   "IANA Registry- Transform Type 3 - Integrity
                    Algorithm Transform IDs. RFC 8221 - Cryptographic
                    Algorithm Implementation Requirements and Usage
                    Guidance for Encapsulating Security Payload (ESP)
                    and Authentication Header (AH) and RFC 8247 -
                    Algorithm Implementation Requirements and Usage
                    Guidance for the Internet Key Exchange Protocol
                    Version 2 (IKEv2).";
           }

           typedef ipsec-mode {
```

```
               type enumeration {
                   enum transport {
                       description
                           "IPsec transport mode. No Network Address
                            Translation (NAT) support.";
                   }
                   enum tunnel {
                       description "IPsec tunnel mode.";
                   }
               }
               description
                   "Type definition of IPsec mode: transport or
                    tunnel.";
               reference
                   "Section 3.2 in RFC 4301.";
           }

           typedef esp-encap {
               type enumeration {
                   enum espintcp {
                       description
                           "ESP in TCP encapsulation.";
                       reference
                           "RFC 8229 - TCP Encapsulation of IKE and
                            IPsec Packets.";
                   }
                   enum espintls {
                       description
                           "ESP in TCP encapsulation using TLS.";
                       reference
                           "RFC 8229 - TCP Encapsulation of IKE and
                            IPsec Packets.";
                   }
                   enum espinudp {
                       description
                           "ESP in UDP encapsulation.";
                       reference
                           "RFC 3948 - UDP Encapsulation of IPsec ESP
                           Packets.";
                   }
                   enum none {
                       description
                           "NOT ESP encapsulation.";
                   }
               }
               description
                   "Types of ESP encapsulation when Network Address
                    Translation (NAT) is present between two NSFs.";
```

```
                reference
                    "RFC 8229 - TCP Encapsulation of IKE and IPsec
                     Packets and RFC 3948 - UDP Encapsulation of IPsec
                     ESP Packets.";
            }

            typedef ipsec-protocol-parameters {
                type enumeration {
                    enum esp { description "IPsec ESP protocol."; }
                }
                description
                    "Only the Encapsulation Security Protocol (ESP) is
                     supported but it could be extended in the future.";
                reference
                    "RFC 4303- IP Encapsulating Security Payload
                    (ESP).";

            }

            typedef lifetime-action {
                type enumeration {
                    enum terminate-clear {
                        description
                            "Terminates the IPsec SA and allows the
                             packets through.";
                    }
                    enum terminate-hold {
                        description
                            "Terminates the IPsec SA and drops the
                             packets.";
                    }
                    enum replace  {
                        description
                            "Replaces the IPsec SA with a new one:
                            rekey. ";
                    }
                }
                description
                    "When the lifetime of an IPsec SA expires an action
                     needs to be performed over the IPsec SA that
                     reached the lifetime. There are three posible
                     options: terminate-clear, terminate-hold and
                     replace.";
                reference
                    "Section 4.5 in RFC 4301.";
            }

            typedef ipsec-traffic-direction {
```

```
               type enumeration {
                   enum inbound {
                       description "Inbound traffic.";
                   }
                   enum outbound {
                       description "Outbound traffic.";
                   }
               }
               description
                   "IPsec traffic direction is defined in two
                    directions: inbound and outbound. From a NSF
                    perspective inbound means the traffic that enters
                    the NSF and outbound is the traffic that is sent
                    from the NSF.";
               reference
                   "Section 5 in RFC 4301.";
           }

           typedef ipsec-spd-action {
               type enumeration {
                   enum protect {
                       description
                           "PROTECT the traffic with IPsec.";
                   }
                   enum bypass {
                       description
                           "BYPASS the traffic. The packet is forwarded
                            without IPsec protection.";
                   }
                   enum discard {
                       description
                           "DISCARD the traffic. The IP packet is
                            discarded.";
                   }
               }
               description
                   "The action when traffic matches an IPsec security
                    policy. According to RFC 4301 there are three
                    possible values: BYPASS, PROTECT AND DISCARD";
               reference
                   "Section 4.4.1 in RFC 4301.";
           }

           typedef ipsec-inner-protocol {
               type union {
                   type uint8;
                   type enumeration {
                       enum any {
```

```
                        value 256;
                        description
                            "Any IP protocol number value.";
                    }
                }
            }
            default any;
            description
                "IPsec protection can be applied to specific IP
                 traffic and layer 4 traffic (TCP, UDP, SCTP, etc.)
                 or ANY protocol in the IP packet payload. We
                 specify the IP protocol number with an uint8 or
                 ANY defining an enumerate with value 256 to
                 indicate the protocol number.";
            reference
                "Section 4.4.1.1 in RFC 4301.
                 IANA Registry - Protocol Numbers.";
        }

        grouping encap {
            description
                "This group of nodes allows to define the type of
                 encapsulation in case NAT traversal is
                 required and port information.";
            leaf espencap {
                type esp-encap;
                description
                    "ESP in TCP, ESP in UDP or ESP in TLS.";
            }
            leaf sport {
                type inet:port-number;
                default 4500;
                description
                    "Encapsulation source port.";
            }
            leaf dport {
                type inet:port-number;
                default 4500;
                description
                    "Encapsulation destination port.";
            }

            leaf-list oaddr {
                type inet:ip-address;
                description
                    "If required, this is the original address that
                     was used before NAT was applied over the Packet.
                     ";
```

```
               }
               reference
                   "RFC 3947 and RFC 8229.";
           }

           grouping lifetime {
               description
                   "Different lifetime values limited to an IPsec SA.";
               leaf time {
                   type uint32;
                   default 0;
                   description
                       "Time in seconds since the IPsec SA was added.
                        For example, if this value is 180 seconds it
                        means the IPsec SA expires in 180 seconds since
                        it was added. The value 0 implies infinite.";
               }
               leaf bytes {
                   type uint32;
                   default 0;
                   description
                       "If the IPsec SA processes the number of bytes
                       expressed in this leaf, the IPsec SA expires and
                       should be rekeyed. The value 0 implies
                       infinite.";
               }
               leaf packets {
                   type uint32;
                   default 0;
                   description
                       "If the IPsec SA processes the number of packets
                       expressed in this leaf, the IPsec SA expires and
                       should be rekeyed. The value 0 implies
                       infinite.";
               }
               leaf idle {
                   type uint32;
                   default 0;
                   description
                       "When a NSF stores an IPsec SA, it
                        consumes system resources. In an idle NSF this
                        is a waste of resources. If the IPsec SA is idle
                        during this number of seconds the IPsec SA
                        should be removed. The value 0 implies
                        infinite.";
               }
               reference
                   "Section 4.4.2.1 in RFC 4301.";
```

```
           }

           grouping port-range  {
               description
                   "This grouping defines a port range, such as
                    expressed in RFC 4301. For example: 1500 (Start
                    Port Number)-1600 (End Port Number). A port range
                    is used in the Traffic Selector.";

               leaf start {
                   type inet:port-number;
                   description
                       "Start port number.";
               }
               leaf end {
                   type inet:port-number;
                   description
                       "End port number.";
               }
               reference "Section 4.4.1.2 in RFC 4301.";
           }

           grouping tunnel-grouping {
               description
                   "The parameters required to define the IP tunnel
                    endpoints when IPsec SA requires tunnel mode. The
                    tunnel is defined by two endpoints: the local IP
                    address and the remote IP address.";

               leaf local {
                   type inet:ip-address;
                   mandatory true;
                   description
                       "Local IP address' tunnel endpoint.";
               }
               leaf remote {
                   type inet:ip-address;
                   mandatory true;
                   description
                       "Remote IP address' tunnel endpoint.";
               }
               leaf df-bit {
                   type enumeration {
                       enum clear {
                           description
                               "Disable the DF (Don't Fragment) bit
                                from the outer header. This is the
                                default value.";
```

```
                       }
                       enum set {
                           description
                               "Enable the DF bit in the outer header.";
                       }
                       enum copy {
                           description
                               "Copy the DF bit to the outer header.";
                       }
                   }
                   default clear;
                   description
                       "Allow configuring the DF bit when encapsulating
                        tunnel mode IPsec traffic. RFC 4301 describes
                        three options to handle the DF bit during
                        tunnel encapsulation: clear, set and copy from
                        the inner IP header.";
                   reference
                       "Section 8.1 in RFC 4301.";
               }
               leaf bypass-dscp {
                   type boolean;
                   default true;
                   description
                       "If DSCP (Differentiated Services Code Point)
                        values in the inner header have to be used to
                        select one IPsec SA among several that match
                        the traffic selectors for an outbound packet";
                   reference
                       "Section 4.4.2.1. in RFC 4301.";
               }
               leaf dscp-mapping {
                   type yang:hex-string;
                   description
                       "DSCP values allowed for packets carried over
                        this IPsec SA.";
                   reference
                       "Section 4.4.2.1. in RFC 4301.";
               }
               leaf ecn {
                   type boolean;
                   default false;
                   description
                       "Explicit Congestion Notification (ECN). If true
                        copy CE bits to inner header.";
                   reference
                       "Section 5.1.2 and Annex C in RFC 4301.";
               }
```

```
           }

           grouping selector-grouping {
               description
                   "This grouping contains the definition of a Traffic
                    Selector, which is used in the IPsec policies and
                    IPsec SAs.";

               leaf local-subnet {
                   type inet:ip-prefix;
                   mandatory true;
                   description
                       "Local IP address subnet.";
               }
               leaf remote-subnet {
                   type inet:ip-prefix;
                   mandatory true;
                   description
                       "Remote IP address subnet.";
               }
               leaf inner-protocol {
                   type ipsec-inner-protocol;
                   default any;
                   description
                       "Inner Protocol that is going to be
                       protected with IPsec.";
               }
               list local-ports {
                   key "start end";
                   uses port-range;
                   description
                       "List of local ports. When the inner
                        protocol is ICMP this 16 bit value represents
                        code and type.";
               }
               list remote-ports {
                   key "start end";
                   uses port-range;
                   description
                       "List of remote ports. When the upper layer
                       protocol is ICMP this 16 bit value represents
                       code and type.";
               }
               reference
                   "Section 4.4.1.2 in RFC 4301.";
           }

           grouping ipsec-policy-grouping {
```

```
               description
                   "Holds configuration information for an IPsec SPD
                    entry.";

               leaf anti-replay-window {
                   type uint64;
                   default 32;
                   description
                       "A 64-bit counter used to determine whether an
                        inbound ESP packet is a replay.";
                   reference
                       "Section 4.4.2.1 in RFC 4301.";
               }
               container traffic-selector {
                   description
                       "Packets are selected for
                        processing actions based on the IP and inner
                        protocol header information, selectors,
                        matched against entries in the SPD.";
                   uses selector-grouping;
                   reference
                       "Section 4.4.4.1 in RFC 4301.";
               }
               container processing-info {
                   description
                       "SPD processing. If the required processing
                        action is protect, it contains the required
                        information to process the packet.";
                   leaf action {
                       type ipsec-spd-action;
                       default discard;
                       description
                           "If bypass or discard, container
                           ipsec-sa-cfg is empty.";
                   }
                   container ipsec-sa-cfg {
                       when "../action = 'protect'";
                       description
                           "IPSec SA configuration included in the SPD
                           entry.";
                       leaf pfp-flag {
                           type boolean;
                           default false;
                           description
                                "Each selector has a Populate From
                                 Packet (PFP) flag. If asserted for a
                                 given selector X, the flag indicates
                                 that the IPSec SA to be created should
```

```
                                  take its value (local IP address,
                                  remote IP address, Next Layer
                                  Protocol, etc.) for X from the value
                                  in the packet. Otherwise, the IPsec SA
                                  should take its value(s) for X from
                                  the value(s) in the SPD entry.";
                       }
                       leaf ext-seq-num {
                           type boolean;
                           default false;
                           description
                                 "True if this IPsec SA is using extended
                                  sequence numbers. True 64 bit counter,
                                  False 32 bit.";
                       }
                       leaf seq-overflow {
                           type boolean;
                           default false;
                           description
                               "The flag indicating whether
                               overflow of the sequence number
                               counter should prevent transmission
                               of additional packets on the IPsec
                               SA (false) and, therefore needs to
                               be rekeyed, or whether rollover is
                               permitted (true). If Authenticated
                               Encryption with Associated Data
                               (AEAD) is used this flag MUST be
                               false.";
                       }
                       leaf stateful-frag-check {
                           type boolean;
                           default false;
                           description
                               "Indicates whether (true) or not (false)
                                stateful fragment checking applies to
                                the IPsec SA to be created.";
                       }
                       leaf mode {
                           type ipsec-mode;
                           default transport;
                           description
                               "IPsec SA has to be processed in
                                transport or tunnel mode.";
                       }
                       leaf protocol-parameters {
                           type ipsec-protocol-parameters;
                           default esp;
```

```
                           description
                                "Security protocol of the IPsec SA:
                                Only ESP is supported but it could be
                                extended in the future.";
                       }
                       container esp-algorithms {
                           when "../protocol-parameters = 'esp'";
                           description
                                "Configuration of Encapsulating
                                Security Payload (ESP) parameters and
                                algorithms.";
                           leaf-list integrity {
                               type integrity-algorithm-type;
                               default 0;
                               ordered-by user;
                               description
                                   "Configuration of ESP authentication
                                   based on the specified integrity
                                   algorithm. With AEAD algorithms,
                                   the integrity node is not
                                   used.";
                               reference
                                   "Section 3.2 in RFC 4303.";
                           }
                           leaf-list encryption {
                               type encryption-algorithm-type;
                               default 20;
                               ordered-by user;
                               description
                                   "Configuration of ESP encryption
                                   algorithms. The default value is
                                   20 (ENCR_AES_GCM_16).";
                               reference
                                   "Section 3.2 in RFC 4303.";
                           }
                           leaf tfc-pad {
                               type boolean;
                               default false;
                               description
                                   "If Traffic Flow Confidentiality
                                    (TFC) padding for ESP encryption
                                    can be used (true) or not (false)";
                               reference
                                   "Section 2.7 in RFC 4303.";
                           }
                           reference
                               "RFC 4303.";
                       }
```

```
                       container tunnel {
                           when "../mode = 'tunnel'";
                           uses tunnel-grouping;
                           description
                              "IPsec tunnel endpoints definition.";
                       }
                   }
                   reference
                       "Section 4.4.1.2 in RFC 4301.";
               }
               container spd-mark {
                       description
                           "The Mark to set for the IPsec SA of this
                            connection. This option is only available
                            on linux NETKEY/XFRM kernels. It can be
                            used with iptables to create custom
                            iptables rules using CONNMARK. It can also
                            be used with Virtual Tunnel Interfaces
                            (VTI) to direct marked traffic to
                            specific vtiXX devices.";
                       leaf mark {
                           type uint32;
                           default 0;
                           description
                               "Mark used to match XFRM policies and
                                states.";
                       }
                       leaf mask {
                           type yang:hex-string;
                           default 00:00:00:00;
                           description
                               "Mask used to match XFRM policies and
                               states.";
                       }
               }
           }
       }

       <CODE ENDS>
```

Appendix B. Appendix B: YANG model for IKE case

```
       <CODE BEGINS> file "ietf-ipsec-ike@2019-08-05.yang"
```

```
       module ietf-ipsec-ike {
           yang-version 1.1;
           namespace "urn:ietf:params:xml:ns:yang:ietf-ipsec-ike";
           prefix "ike";

           import ietf-inet-types { prefix inet; }
           import ietf-yang-types { prefix yang; }

           import ietf-crypto-types {
               prefix ct;
               reference
                   "draft-ietf-netconf-crypto-types-10:
                   Common YANG Data Types for Cryptography.";
           }

           import ietf-ipsec-common {
               prefix ic;
               reference
                   "RFC XXXX: module ietf-ipsec-common, revision
                    2019-08-05.";
           }

           import ietf-netconf-acm {
                  prefix nacm;
                  reference
                    "RFC 8341: Network Configuration Access Control
                     Model.";
           }

           organization "IETF I2NSF Working Group";

           contact
           "WG Web:  <https://datatracker.ietf.org/wg/i2nsf/about/>
            WG List: <mailto:i2nsf@ietf.org>

           Author: Rafael Marin-Lopez
                   <mailto:rafa@um.es>

           Author: Gabriel Lopez-Millan
                   <mailto:gabilm@um.es>

           Author: Fernando Pereniguez-Garcia
                   <mailto:fernando.pereniguez@cud.upct.es>
           ";

           description

           "This module contains IPSec IKE case model for the SDN-based
```

```
            IPsec flow protection service. An NSF will implement this
            module.


           Copyright (c) 2019 IETF Trust and the persons identified as
           authors of the code.  All rights reserved.

           Redistribution and use in source and binary forms, with or
           without modification, is permitted pursuant to, and subject
           to the license terms contained in, the Simplified BSD License
           set forth in Section 4.c of the IETF Trust's Legal Provisions
           Relating to IETF Documents
           (http://trustee.ietf.org/license-info).

           This version of this YANG module is part of RFC XXXX; see
           the RFC itself for full legal notices.

           The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
           'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
           'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this
           document are to be interpreted as described in BCP 14
           (RFC 2119) (RFC 8174) when, and only when, they appear
           in all capitals, as shown here.";

           revision "2019-08-05" {
               description "Revision 6";
               reference
                   "RFC XXXX: YANG model for IKE case.";
           }

           typedef ike-spi {
               type uint64 { range "0..max"; }
               description
                   "Security Parameter Index (SPI)'s IKE SA.";
               reference
                   "Section 2.6 in RFC 7296.";
           }

           typedef autostartup-type {
               type enumeration {
                   enum add {
                       description
                           "IKE/IPsec configuration is only loaded into
                            IKE implementation but IKE/IPsec SA is not
                            started.";
                   }
                   enum on-demand {
                       description
```

```
                   "IKE/IPsec configuration is loaded
                   into IKE implementation. The IPsec policies
                   are transferred to the NSF's kernel but the
                   IPsec SAs are not established immediately.
                   The IKE implementation will negotiate the
                   IPsec SAs when the NSF's kernel requests it
                   (i.e. through an ACQUIRE notification).";
           }
           enum start {
               description "IKE/IPsec configuration is loaded
               and transferred to the NSF's kernel, and the
               IKEv2 based IPsec SAs are established
               immediately without waiting any packet.";
           }
       }
       description
           "Different policies to set IPsec SA configuration
            into NSF's kernel when IKEv2 implementation has
            started.";
   }



   typedef pfs-group {
       type uint16;
       description
           "DH groups for IKE and IPsec SA rekey.";
       reference
           "Section 3.3.2 in RFC 7296. Transform Type 4 -
            Diffie-Hellman Group Transform IDs in IANA Registry
             - Internet Key Exchange Version 2 (IKEv2)
            Parameters.";
   }

   typedef auth-protocol-type {
       type enumeration {
           enum ikev2 {
               value 2;
               description
                   "IKEv2 authentication protocol. It is the
                    only defined right now. An enum is used for
                    further extensibility.";
           }
       }
       description
           "IKE authentication protocol version specified in the
            Peer Authorization Database (PAD). It is defined as
            enumerate to allow new IKE versions in the
```

```
                    future.";
               reference
                   "RFC 7296.";
           }

           typedef auth-method-type {
               type enumeration {
                   enum pre-shared {
                       description
                           "Select pre-shared key as the
                           authentication method.";
                       reference
                           "RFC 7296.";
                   }
                   enum eap {
                       description
                           "Select EAP as the authentication method.";
                       reference
                           "RFC 7296.";
                   }
                   enum digital-signature {
                       description
                           "Select digital signature method.";
                       reference
                           "RFC 7296 and RFC 7427.";
                   }
                   enum null {
                       description
                           "Null authentication.";
                       reference
                           "RFC 7619.";
                   }

               }
               description
                   "Peer authentication method specified in the Peer
                    Authorization Database (PAD).";
           }

           container ipsec-ike {
               description
                   "IKE configuration for a NSF. It includes PAD
                    parameters, IKE connections information and state
                    data.";

               container pad {
                   description
                      "Configuration of Peer Authorization Database
```

```
                    (PAD). The PAD contains information about IKE
                    peer (local and remote). Therefore, the Security
                    Controller also stores authentication
                    information for this NSF and can include
                    several entries for the local NSF not only
                    remote peers. Storing local and remote
                    information makes possible to specify that this
                    NSF with identity A will use some particular
                    authentication with remote NSF with identity B
                    and what are the authentication mechanisms
                    allowed to B.";
                list pad-entry {
                    key "name";
                    ordered-by user;
                    description
                        "Peer Authorization Database (PAD) entry. It
                         is a list of PAD entries ordered by the
                         Security Controller.";
                    leaf name {
                        type string;
                        description
                            "PAD unique name to identify this
                             entry.";
                    }
                    choice identity {
                        mandatory true;
                        description
                            "A particular IKE peer will be
                            identified by one of these identities.
                            This peer can be a remote peer or local
                            peer (this NSF).";
                        reference
                            "Section 4.4.3.1 in RFC 4301.";
                        case ipv4-address{
                            leaf ipv4-address {
                                type inet:ipv4-address;
                                description
                                    "Specifies the identity as a
                                     single four (4) octet.";
                            }
                        }
                        case ipv6-address{
                            leaf ipv6-address {
                                type inet:ipv6-address;
                                description
                                    "Specifies the identity as a
                                     single sixteen (16) octet IPv6
                                     address. An example is
```

```
                                2001:DB8:0:0:8:800:200C:417A.";
                        }
                    }
                    case fqdn-string {
                        leaf fqdn-string {
                            type inet:domain-name;
                            description
                                "Specifies the identity as a
                                 Fully-QualifiedDomain Name
                                 (FQDN) string. An example is:
                                 example.com. The string MUST
                                 NOT contain any terminators
                                 (e.g., NULL, CR, etc.).";
                        }
                    }
                    case rfc822-address-string {
                        leaf rfc822-address-string {
                            type string;
                            description
                                "Specifies the identity as a
                                 fully-qualified RFC822 email
                                 address string. An example is,
                                 jsmith@example.com. The string
                                 MUST NOT contain any
                                 terminators e.g., NULL, CR,
                                 etc.).";
                            reference
                                "RFC 822.";
                        }
                    }
                    case dnx509 {
                        leaf dnx509 {
                            type string;
                            description
                                "Specifies the identity as a
                                 ASN.1 X.500 Distinguished
                                 Name. An example is
                                 C=US,O=Example
                                 Organisation,CN=John Smith.";
                            reference
                                "RFC 2247.";
                        }
                    }
                    case gnx509 {
                        leaf gnx509 {
                            type string;
                            description
                                "ASN.1 X.509 GeneralName. RFC
```

```
                                       3280.";
                              }
                          }
                          case id-key {
                              leaf id-key {
                                  type string;
                                  description
                                      "Opaque octet stream that may be
                                       used to pass vendor-specific
                                       information for proprietary
                                       types of identification.";
                                  reference
                                      "Section 3.5 in RFC 7296.";
                              }
                          }
                          case id-null {
                              leaf id-null {
                                  type empty;
                                  description
                                      "ID_NULL identification used
                                       when IKE identification payload
                                       is not used." ;
                                  reference
                                      "RFC 7619.";
                              }
                          }
                      }
                      leaf auth-protocol {
                          type auth-protocol-type;
                          default ikev2;
                          description
                              "Only IKEv2 is supported right now but
                               other authentication protocols may be
                               supported in the future.";
                      }
                      container peer-authentication {
                          description
                              "This container allows the Security
                               Controller to configure the
                               authentication method (pre-shared key,
                               eap, digitial-signature, null) that
                               will use a particular peer and the
                               credentials, which will depend on the
                               selected authentication method.";
                          leaf auth-method {
                             type auth-method-type;
                             default pre-shared;
                             description
```

```
                                   "Type of authentication method
                                   (pre-shared, eap, digital signature,
                                    null).";
                              reference
                                  "Section 2.15 in RFC 7296.";
                           }
                           container eap-method {
                               when "../auth-method = 'eap'";
                               leaf eap-type {
                                   type uint8;
                                   mandatory true;
                                   description
                                       "EAP method type. This
                                       information provides the
                                       particular EAP method to be
                                       used. Depending on the EAP
                                       method, pre-shared keys or
                                       certificates may be used.";
                               }
                               description
                                   "EAP method description used when
                                   authentication method is 'eap'.";
                               reference
                                   "Section 2.16 in RFC 7296.";
                           }
                           container pre-shared {
                               when
                                   "../auth-method[.='pre-shared' or
                                    .='eap']";
                               leaf secret {
                                   nacm:default-deny-all;
                                   type yang:hex-string;
                                   description
                                       "Pre-shared secret value. The
                                        NSF has to prevent read access
                                        to this value for security
                                        reasons.";
                               }
                               description
                                   "Shared secret value for PSK or
                                    EAP method authentication based on
                                    PSK.";
                           }
                           container digital-signature {
                               when
                                "../auth-method[.='digital-signature'
                               or .='eap']";
                               leaf ds-algorithm {
```

```
                                    type uint8;
                                    description
                                        "The digital signature
                                        algorithm is specified with a
                                        value extracted from the IANA
                                        Registry. Depending on the
                                        algorithm, the following leafs
                                        must contain information. For
                                        example if digital signature
                                        involves a certificate then leaf
                                        'cert-data' and 'private-key'
                                        will contain this information.";
                                    reference
                                        "IKEv2 Authentication Method -
                                         IANA Registry - Internet Key
                                         Exchange Version 2 (IKEv2)
                                         Parameters.";
                                }

                                choice public-key {
                                    mandatory true;
                                    leaf raw-public-key {
                                        type binary;
                                        description
                                          "A binary that contains the
                                          value of the public key.  The
                                          interpretation of the content
                                          is defined by the digital
                                          signature algorithm. For
                                          example, an RSA key is
                                          represented as RSAPublicKey as
                                          defined in RFC 8017, and an
                                          Elliptic Curve Cryptography
                                          (ECC) key is represented
                                          using the 'publicKey'
                                          described in RFC 5915.";
                                    reference
                                          "RFC XXX: Common YANG Data
                                          Types for Cryptography.";
                                    }
                                    leaf cert-data {
                                        type ct:x509;
                                        description
                                            "X.509 certificate data -
                                             PEM4.";
                                        reference
                                            "RFC XXX: Common YANG Data
                                            Types for Cryptography.";
```

```
                                   }
                                   description
                                       "If the Security Controller
                                        knows that the NSF
                                        already owns a private key
                                        associated to this public key
                                        (the NSF generated the pair
                                        public key/private key out of
                                        band), it will only configure
                                        one of the leaf of this
                                        choice. The NSF, based on
                                        the public key value can know
                                        the private key to be used.";
                               }
                               leaf private-key {
                                   nacm:default-deny-all;
                                   type binary;
                                   description
                                       "A binary that contains the
                                        value of the private key. The
                                        interpretation of the content
                                        is defined by the digital
                                        signature algorithm. For
                                        example, an RSA key is
                                        represented as RSAPrivateKey as
                                        defined in RFC 8017, and an
                                        Elliptic Curve Cryptography
                                        (ECC) key is represented as
                                        ECPrivateKey as defined in RFC
                                        5915.";
                                   reference
                                       "RFC XXX: Common YANG Data
                                       Types for Cryptography.";
                               }
                               leaf-list ca-data {
                                   type ct:x509;
                                   description
                                       "List of trusted Certification
                                       Authorities (CA) certificates
                                       encoded using ASN.1
                                       distinguished encoding rules
                                       (DER).";
                                   reference
                                       "RFC XXX: Common YANG Data
                                       Types for Cryptography.";
                               }
                               leaf crl-data {
                                   type ct:crl;
```

```
                                      description
                                         "A CertificateList structure, as
                                          specified in RFC 5280,
                                          encoded using ASN.1
                                          distinguished encoding rules
                                          (DER),as specified in ITU-T
                                          X.690.";
                                      reference
                                          "RFC XXX: Common YANG Data Types
                                           for Cryptography.";
                                  }
                                  leaf crl-uri  {
                                      type inet:uri;
                                      description
                                          "X.509 CRL certificate URI.";
                                  }
                                  leaf oscp-uri {
                                      type inet:uri;
                                      description
                                          "OCSP URI.";
                                  }
                                  description
                                      "Digital Signature container.";

                              } /*container digital-signature*/
                          } /*container peer-authentication*/
                      }
                  }

                  list conn-entry {
                      key "name";
                      description
                          "IKE peer connection information. This list
                          contains the IKE connection for this peer
                          with other peers. This will be translated in
                          real time by IKE Security Associations
                          established with these nodes.";
                      leaf name {
                          type string;
                          mandatory true;
                          description
                              "Identifier for this connection
                               entry.";
                      }
                      leaf autostartup {
                            type autostartup-type;
                            default add;
                            description
```

```
                             "By-default: Only add configuration
                              without starting the security
                              association.";
                   }
                   leaf initial-contact {
                       type boolean;
                       default false;
                       description
                           "The goal of this value is to deactivate the
                           usage of INITIAL_CONTACT notification
                           (true). If this flag remains to false it
                           means the usage of the INITIAL_CONTACT
                           notification will depend on the IKEv2
                           implementation.";
                   }
                   leaf version {
                       type auth-protocol-type;
                       default ikev2;
                       description
                          "IKE version. Only version 2 is supported
                          so far.";
                   }
                   leaf fragmentation {
                       type boolean;
                       default false;
                       description
                           "Whether or not to enable IKE
                            fragmentation as per RFC 7383 (true or
                            false).";
                       reference
                           "RFC 7383.";
                   }
                   container ike-sa-lifetime-soft {
                       description
                           "IKE SA lifetime soft. Two lifetime values
                            can be configured: either rekey time of the
                            IKE SA or reauth time of the IKE SA. When
                            the rekey lifetime expires a rekey of the
                            IKE SA starts. When reauth lifetime
                            expires a IKE SA reauthentication starts.";
                      leaf rekey-time {
                           type uint32;
                           default 0;
                           description
                               "Time in seconds between each IKE SA
                               rekey.The value 0 means infinite.";
                      }
                      leaf reauth-time {
```

```
                           type uint32;
                           default 0;
                           description
                             "Time in seconds between each IKE SA
                             reauthentication. The value 0 means
                             infinite.";
                      }
                      reference
                          "Section 2.8 in RFC 7296.";
                   }
                   container ike-sa-lifetime-hard {
                       description
                           "Hard IKE SA lifetime. When this
                            time is reached the IKE SA is removed.";
                       leaf over-time {
                           type uint32;
                           default 0;
                           description
                               "Time in seconds before the IKE SA is
                                removed. The value 0 means infinite.";
                       }
                       reference
                           "RFC 7296.";
                   }
                   leaf-list authalg {
                       type ic:integrity-algorithm-type;
                       default 12;
                       ordered-by user;
                       description
                          "Authentication algorithm for establishing
                          the IKE SA. This list is ordered following
                          from the higher priority to lower priority.
                          First node of the list will be the algorithm
                          with higher priority. If this list is empty
                          the default integrity algorithm value assumed
                          is NONE.";
                   }
                   leaf-list encalg {
                       type ic:encryption-algorithm-type;
                       default 12;
                       ordered-by user;
                       description
                          "Encryption or AEAD algorithm for the IKE
                          SAs. This list is ordered following
                          from the higher priority to lower priority.
                          First node of the list will be the algorithm
                          with higher priority. If this list is empty
                          the default encryption value assumed is
```

```
                          NULL.";
                   }
                   leaf dh-group {
                       type pfs-group;
                       default 14;
                       description
                           "Group number for Diffie-Hellman
                           Exponentiation used during IKE_SA_INIT
                           for the IKE SA key exchange.";
                   }
                   leaf half-open-ike-sa-timer {
                       type uint32;
                       description
                           "Set the half-open IKE SA timeout
                            duration.";
                       reference
                           "Section 2 in RFC 7296.";
                   }

                   leaf half-open-ike-sa-cookie-threshold {
                       type uint32;
                       description
                           "Number of half-open IKE SAs that activate
                            the cookie mechanism." ;
                       reference
                           "Section 2.6 in RFC 7296.";
                   }
                   container local {
                       leaf local-pad-entry-name {
                           type string;
                           description
                               "Local peer authentication information.
                                This node points to a specific entry in
                                the PAD where the authorization
                                information about this particular local
                                peer is stored. It MUST match a
                                pad-entry-name.";
                       }
                       description
                           "Local peer authentication information.";
                   }
                   container remote {
                       leaf remote-pad-entry-name {
                           type string;
                           description
                               "Remote peer authentication information.
                                This node points to a specific entry in
                                the PAD where the authorization
```

```
                                information about this particular
                                remote peer is stored. It MUST match a
                                pad-entry-name.";
                       }
                       description
                           "Remote peer authentication information.";
                   }
                   container encapsulation-type
                   {
                       uses ic:encap;
                       description
                           "This container carries configuration
                           information about the source and destination
                           ports of encapsulation that IKE should use
                           and the type of encapsulation that
                           should use when NAT traversal is required.
                           However, this is just a best effort since
                           the IKE implementation may need to use a
                           different encapsulation as
                           described in RFC 8229.";
                       reference
                           "RFC 8229.";
                   }
                   container spd {
                       description
                           "Configuration of the Security Policy
                           Database (SPD). This main information is
                           placed in the grouping
                           ipsec-policy-grouping.";
                       list spd-entry {
                           key "name";
                           ordered-by user;
                           leaf name {
                               type string;
                               mandatory true;
                               description
                                   "SPD entry unique name to identify
                                   the IPsec policy.";
                           }
                           container ipsec-policy-config {
                               description
                                   "This container carries the
                                   configuration of a IPsec policy.";
                               uses ic:ipsec-policy-grouping;
                           }
                           description
                               "List of entries which will constitute
                               the representation of the SPD. Since we
```

```
                               have IKE in this case, it is only
                               required to send a IPsec policy from
                               this NSF where 'local' is this NSF and
                               'remote' the other NSF. The IKE
                               implementation will install IPsec
                               policies in the NSF's kernel in both
                               directions (inbound and outbound) and
                               their corresponding IPsec SAs based on
                               the information in this SPD entry.";
                       }
                       reference
                           "Section 2.9 in RFC 7296.";
                   }
                   container child-sa-info {
                       leaf-list pfs-groups {
                           type pfs-group;
                           default 0;
                           ordered-by user;
                           description
                               "If non-zero, it is required perfect
                                forward secrecy when requesting new
                                IPsec SA. The non-zero value is
                                the required group number. This list is
                                ordered following from the higher
                                priority to lower priority. First node
                                of the list will be the algorithm
                                with higher priority.";
                       }
                       container child-sa-lifetime-soft {
                           description
                               "Soft IPsec SA lifetime soft.
                                After the lifetime the action is
                                defined in this container
                                in the leaf action.";
                           uses ic:lifetime;
                           leaf action {
                               type ic:lifetime-action;
                               default replace;
                               description
                                   "When the lifetime of an IPsec SA
                                    expires an action needs to be
                                    performed over the IPsec SA that
                                    reached the lifetime. There are
                                    three possible options:
                                    terminate-clear, terminate-hold and
                                    replace.";
                           reference
                               "Section 4.5 in RFC 4301 and Section 2.8
```

```
                                in RFC 7296.";
                           }
                       }
                       container child-sa-lifetime-hard {
                           description
                               "IPsec SA lifetime hard. The action will
                                be to terminate the IPsec SA.";
                           uses ic:lifetime;
                           reference
                               "Section 2.8 in RFC 7296.";
                       }
                       description
                           "Specific information for IPsec SAs
                           SAs. It includes PFS group and IPsec SAs
                           rekey lifetimes.";
                   }
                   container state {
                       config false;

                       leaf initiator {
                           type boolean;
                           description
                               "It is acting as initiator for this
                                connection.";
                       }
                       leaf initiator-ikesa-spi {
                           type ike-spi;
                           description
                               "Initiator's IKE SA SPI.";
                       }
                       leaf responder-ikesa-spi {
                           type ike-spi;
                           description
                               "Responder's IKE SA SPI.";
                       }
                       leaf nat-local {
                           type boolean;
                           description
                               "True, if local endpoint is behind a
                                NAT.";
                       }
                       leaf nat-remote {
                           type boolean;
                           description
                               "True, if remote endpoint is behind
                               a NAT.";
                       }
```

```
                       container encapsulation-type
                       {
                           uses ic:encap;
                           description
                               "This container provides information
                               about the source and destination
                               ports of encapsulation that IKE is
                               using, and the type of encapsulation
                               when NAT traversal is required.";
                           reference
                               "RFC 8229.";
                       }
                       leaf established {
                           type uint64;
                           description
                               "Seconds since this IKE SA has been
                                established.";
                       }
                       leaf current-rekey-time {
                           type uint64;
                           description
                               "Seconds before IKE SA must be rekeyed.";
                       }
                       leaf current-reauth-time {
                           type uint64;
                           description
                               "Seconds before IKE SA must be
                                re-authenticated.";
                       }
                       description
                           "IKE state data for a particular
                            connection.";
                   } /* ike-sa-state */
               } /* ike-conn-entries */

               container number-ike-sas {
                   config false;
                   leaf total {
                       type uint64;
                       description
                           "Total number of active IKE SAs.";
                   }
                   leaf half-open {
                       type uint64;
                       description
                           "Number of half-open active IKE SAs.";
                   }
                   leaf half-open-cookies {
```

```
                       type uint64;
                       description
                           "Number of half open active IKE SAs with
                            cookie activated.";
                   }
                   description
                       "General information about the IKE SAs. In
                       particular, it provides the current number of
                       IKE SAs.";
               }
           }  /* container ipsec-ike */
       }


       <CODE ENDS>
```

Appendix C. Appendix C: YANG model for IKE-less case

```
       <CODE BEGINS> file "ietf-ipsec-ikeless@2019-08-05.yang"

       module ietf-ipsec-ikeless {

           yang-version 1.1;
           namespace "urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless";

           prefix "ikeless";

           import ietf-yang-types { prefix yang; }

           import ietf-ipsec-common {
               prefix ic;
               reference
                   "Common Data model for SDN-based IPSec
                    configuration.";
           }

           import ietf-netconf-acm {
                  prefix nacm;
                  reference
                    "RFC 8341: Network Configuration Access Control
                     Model.";
           }
```

```
           organization "IETF I2NSF Working Group";

           contact
           "WG Web:  <https://datatracker.ietf.org/wg/i2nsf/about/>
            WG List: <mailto:i2nsf@ietf.org>

           Author: Rafael Marin-Lopez
                   <mailto:rafa@um.es>

           Author: Gabriel Lopez-Millan
                   <mailto:gabilm@um.es>

           Author: Fernando Pereniguez-Garcia
                   <mailto:fernando.pereniguez@cud.upct.es>
           ";

           description
               "Data model for IKE-less case in the SDN-base IPsec flow
                protection service.

                Copyright (c) 2019 IETF Trust and the persons
                identified as authors of the code.  All rights reserved.
                Redistribution and use in source and binary forms, with
                or without modification, is permitted pursuant to, and
                subject to the license terms contained in, the
                Simplified BSD License set forth in Section 4.c of the
                IETF Trust's Legal Provisions Relating to IETF Documents
                (https://trustee.ietf.org/license-info).

                This version of this YANG module is part of RFC XXXX;;
                see the RFC itself for full legal notices.

                The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
                'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
                'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this
                document are to be interpreted as described in BCP 14
                (RFC 2119) (RFC 8174) when, and only when, they appear
                in all capitals, as shown here.";


           revision "2019-08-05" {
               description "Revision 06";
               reference "RFC XXXX: YANG model for IKE case.";
           }


           container ipsec-ikeless {
               description
```

```
                   "Container for configuration of the IKE-less
                    case. The container contains two additional
                    containers: 'spd' and 'sad'. The first allows the
                    Security Controller to configure IPsec policies in
                    the Security Policy Database SPD, and the second
                    allows to configure IPsec Security Associations
                    (IPsec SAs) in the Security Association Database
                    (SAD).";
               reference "RFC 4301.";
               container spd {
                   description
                       "Configuration of the Security Policy Database
                        (SPD.)";
                   reference "Section 4.4.1.2 in RFC 4301.";

                   list spd-entry {
                       key "name";
                       ordered-by user;
                       leaf name {
                           type string;
                           mandatory true;
                           description
                               "SPD entry unique name to identify this
                                entry.";
                       }
                       leaf direction {
                           type ic:ipsec-traffic-direction;
                           description
                               "Inbound traffic or outbound
                                traffic. In the IKE-less case the
                                Security Controller needs to
                                specify the policy direction to be
                                applied in the NSF. In the IKE case
                                this direction does not need to be
                                specified since IKE
                                will determine the direction that
                                IPsec policy will require.";
                       }
                       leaf reqid {
                           type uint64;
                           default 0;
                           description
                               "This value allows to link this
                                IPsec policy with IPsec SAs with the
                                same reqid. It is only required in
                                the IKE-less model since, in the IKE
                                case this link is handled internally
                                by IKE.";
```

```
                       }

                       container ipsec-policy-config {
                           description
                               "This container carries the
                               configuration of a IPsec policy.";
                           uses ic:ipsec-policy-grouping;
                       }
                       description
                           "The SPD is represented as a list of SPD
                            entries, where each SPD entry represents an
                            IPsec policy.";
                   } /*list spd-entry*/
               } /*container spd*/

               container sad {
                   description
                       "Configuration of the IPSec Security Association
                        Database (SAD)";
                   reference "Section 4.4.2.1 in RFC 4301.";
                   list sad-entry {
                       key "name";
                       ordered-by user;
                       leaf name {
                           type string;
                           description
                               "SAD entry unique name to identify this
                                entry.";
                       }
                       leaf reqid {
                           type uint64;
                           default 0;
                           description
                               "This value allows to link this
                                IPsec SA with an IPsec policy with
                                the same reqid.";
                       }

                       container ipsec-sa-config {
                           description
                               "This container allows configuring
                               details of an IPsec SA.";
                           leaf spi {
                               type uint32 { range "0..max"; }
                               mandatory true;
                               description
                                   "Security Parameter Index (SPI)'s
                                    IPsec SA.";
```

```
                           }
                           leaf ext-seq-num {
                               type boolean;
                               default true;
                               description
                                   "True if this IPsec SA is using
                                    extended sequence numbers. True 64
                                    bit counter, FALSE 32 bit.";
                           }
                           leaf seq-number-counter {
                               type uint64;
                               default 0;
                               description
                                    "A 64-bit counter when this IPsec
                                    SA is using Extended Sequence
                                    Number or 32-bit counter when it
                                    is not. It used to generate the
                                    initial Sequence Number field
                                    in ESP headers.";
                           }
                           leaf seq-overflow {
                               type boolean;
                               default false;
                               description
                                   "The flag indicating whether
                                    overflow of the sequence number
                                    counter should prevent transmission
                                    of additional packets on the IPsec
                                    SA (false) and, therefore needs to
                                    be rekeyed, or whether rollover is
                                    permitted (true). If Authenticated
                                    Encryption with Associated Data
                                    (AEAD) is used this flag MUST BE
                                    false.";
                           }
                           leaf anti-replay-window {
                               type uint32;
                               default 32;
                               description
                                   "A 32-bit counter and a bit-map (or
                                    equivalent) used to determine
                                    whether an inbound ESP packet is a
                                    replay. If set to 0 no anti-replay
                                    mechanism is performed.";
                           }
                           container traffic-selector {
                               uses ic:selector-grouping;
                               description
```

```
                                   "The IPsec SA traffic selector.";
                           }
                           leaf protocol-parameters {
                               type ic:ipsec-protocol-parameters;
                               default esp;
                               description
                                   "Security protocol of IPsec SA: Only
                                   ESP so far.";
                           }
                           leaf mode {
                               type ic:ipsec-mode;
                               description
                                   "Tunnel or transport mode.";
                           }
                           container esp-sa {
                               when "../protocol-parameters =
                            'esp'";
                               description
                                   "In case the IPsec SA is
                                    Encapsulation Security Payload
                                    (ESP), it is required to specify
                                    encryption and integrity
                                    algorithms, and key material.";

                               container encryption {
                                   description
                                       "Configuration of encryption or
                                        AEAD algorithm for IPSec
                                        Encapsulation Security Payload
                                        (ESP).";

                                   leaf encryption-algorithm {
                                     type ic:encryption-algorithm-type;
                                     description
                                           "Configuration of ESP
                                            encryption. With AEAD
                                            algorithms, the integrity
                                            node is not used.";
                                   }

                                   leaf key {
                                       nacm:default-deny-all;
                                       type yang:hex-string;
                                       description
                                           "ESP encryption key value.";
                                    }
                                   leaf iv {
                                       nacm:default-deny-all;
```

```
                                       type yang:hex-string;
                                       description
                                           "ESP encryption IV value.";
                                   }
                               }
                               container integrity {
                                   description
                                       "Configuration of integrity for
                                        IPSec Encapsulation Security
                                        Payload (ESP). This container
                                        allows to configure integrity
                                        algorithm when no AEAD
                                        algorithms are used, and
                                        integrity is required.";
                                    leaf integrity-algorithm {
                                       type ic:integrity-algorithm-type;
                                       description
                                           "Message Authentication Code
                                           (MAC) algorithm to provide
                                           integrity in ESP.";
                                   }
                                   leaf key {
                                       nacm:default-deny-all;
                                       type yang:hex-string;
                                       description
                                           "ESP integrity key value.";
                                   }
                               }
                           } /*container esp-sa*/

                           container sa-lifetime-hard {
                               description
                                   "IPsec SA hard lifetime. The action
                                   associated is terminate and
                                   hold.";
                               uses ic:lifetime;
                           }
                           container sa-lifetime-soft {
                               description
                                   "IPSec SA soft lifetime.";
                               uses ic:lifetime;
                               leaf action {
                                   type ic:lifetime-action;
                                   description
                                       "Action lifetime:
                                        terminate-clear,
                                        terminate-hold or replace.";
                               }
```

```
                           }
                           container tunnel {
                               when "../mode = 'tunnel'";
                               uses ic:tunnel-grouping;
                               description
                                    "Endpoints of the IPsec tunnel.";
                           }
                           container encapsulation-type
                           {
                               uses ic:encap;
                               description
                                   "This container carries
                                    configuration information about
                                    the source and destination ports
                                    which will be used for ESP
                                    encapsulation that ESP packets the
                                    type of encapsulation when NAT
                                    traversal is in place.";
                           }
                       } /*ipsec-sa-config*/

                       container ipsec-sa-state {
                           config false;
                           description
                               "Container describing IPsec SA state
                               data.";
                           container sa-lifetime-current {
                               uses ic:lifetime;
                               description
                                   "SAD lifetime current.";
                           }
                           container replay-stats {
                               description
                                   "State data about the anti-replay
                                    window.";
                               leaf replay-window {
                                   type uint64;
                                   description
                                       "Current state of the replay
                                        window.";
                               }
                               leaf packet-dropped {
                                   type uint64;
                                   description
                                       "Packets detected out of the
                                        replay window and dropped
                                        because they are replay
                                        packets.";
```

```
                              }
                              leaf failed {
                                  type uint32;
                                  description
                                      "Number of packets detected out
                                       of the replay window.";
                              }
                              leaf seq-number-counter {
                                  type uint64;
                                  description
                                      "A 64-bit counter when this
                                       IPsec SA is using Extended
                                       Sequence Number or 32-bit
                                       counter when it is not.
                                       Current value of sequence
                                       number.";
                              }
                          } /* container replay-stats*/
                      } /*ipsec-sa-state*/

                      description
                          "List of SAD entries that conforms the SAD.";
                  } /*list sad-entry*/
              } /*container sad*/
          }/*container ipsec-ikeless*/

          /* Notifications */
          notification sadb-acquire {
              description
                  "An IPsec SA is required. The traffic-selector
                   container contains information about the IP packet
                   that triggers the acquire notification.";
              leaf ipsec-policy-name {
                  type string;
                  mandatory true;
                  description
                      "It contains the SPD entry name (unique) of
                       the IPsec policy that hits the IP packet
                       required IPsec SA. It is assumed the
                       Security Controller will have a copy of the
                       information of this policy so it can
                       extract all the information with this
                       unique identifier. The type of IPsec SA is
                       defined in the policy so the Security
                       Controller can also know the type of IPsec
                       SA that must be generated.";
              }
              container traffic-selector {
```

```
                    description
                        "The IP packet that triggered the acquire
                         and requires an IPsec SA. Specifically it
                         will contain the IP source/mask and IP
                         destination/mask; protocol (udp, tcp,
                         etc...); and source and destination
                         ports.";
                    uses ic:selector-grouping;
                }
           }

           notification sadb-expire {
               description "An IPsec SA expiration (soft or hard).";
               leaf ipsec-sa-name {
                   type string;
                   mandatory true;
                   description
                       "It contains the SAD entry name (unique) of
                        the IPsec SA that has expired.  It is assumed
                        the Security Controller will have a copy of the
                        IPsec SA information (except the cryptographic
                        material and state data) indexed by this name
                        (unique identifier) so it can know all the
                        information (crypto algorithms, etc.) about
                        the IPsec SA that has expired in order to
                        perform a rekey (soft lifetime) or delete it
                        (hard lifetime) with this unique identifier.";
               }
               leaf soft-lifetime-expire {
                   type boolean;
                   default true;
                   description
                       "If this value is true the lifetime expired is
                        soft. If it is false is hard.";
               }
               container lifetime-current {
                   description
                       "IPsec SA current lifetime. If
                        soft-lifetime-expired is true this container is
                        set with the lifetime information about current
                        soft lifetime.";
                   uses ic:lifetime;
               }
           }
           notification sadb-seq-overflow {
               description "Sequence overflow notification.";
               leaf ipsec-sa-name {
                   type string;
```

```
                   mandatory true;
                   description
                       "It contains the SAD entry name (unique) of
                        the IPsec SA that is about to have sequence
                        number overflow and rollover is not permitted.
                        It is assumed the Security Controller will have
                        a copy of the IPsec SA information (except the
                        cryptographic material and state data) indexed
                        by this name (unique identifier) so the it can
                        know all the information (crypto algorithms,
                        etc.) about the IPsec SA that has expired in
                        order to perform a rekey of the IPsec SA.";
               }
           }
           notification sadb-bad-spi {
               description
                   "Notify when the NSF receives a packet with an
                    incorrect SPI (i.e. not present in the SAD).";
               leaf spi {
                   type uint32 { range "0..max"; }
                   mandatory true;
                   description
                       "SPI number contained in the erroneous IPsec
                        packet.";
               }
           }
       }/*module ietf-ipsec*/

       <CODE ENDS>
```

Appendix D. Example of IKE case, tunnel mode (gateway-to-gateway) with X.509 certificate authentication.

This example shows a XML configuration file sent by the Security Controller to establish a IPsec Security Association between two NSFs in tunnel mode (gateway-to-gateway) with ESP, and authentication based on X.509 certificates using IKEv2.

```
                     Security Controller
                           |
              /---- Southbound interface -----\
             /                                 \
            /                                   \
           /                                     \
          /                                       \
         nsf_h1                                  nsf_h2
 h1---- (:1/:100)===== IPsec_ESP_Tunnel_mode =====(:200/:1)-------h2
 2001:DB8:1:/64       (2001:DB8:123:/64)           2001:DB8:2:/64
```

Figure 7: IKE case, tunnel mode , X.509 certicate authentication.

```
<ipsec-ike xmlns="urn:ietf:params:xml:ns:yang:ietf-ipsec-ike"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <pad>
    <pad-entry>
      <name>nsf_h1_pad</name>
      <ipv6-address>2001:DB8:123::100</ipv6-address>
      <peer-authentication>
         <auth-method>digital-signature</auth-method>
         <digital-signature>
            <cert-data>base64encodedvalue==</cert-data>
            <private-key>base64encodedvalue==</private-key>
            <ca-data>base64encodedvalue==</ca-data>
         </digital-signature>
      </peer-authentication>
    </pad-entry>
    <pad-entry>
      <name>nsf_h2_pad</name>
      <ipv6-address>2001:DB8:123::200</ipv6-address>
      <auth-protocol>ikev2</auth-protocol>
      <peer-authentication>
        <auth-method>digital-signature</auth-method>
        <digital-signature>
          <!-- RSA Digital Signature -->
          <ds-algorithm>1</ds-algorithm>
          <cert-data>base64encodedvalue==</cert-data>
          <ca-data>base64encodedvalue==</ca-data>
        </digital-signature>
      </peer-authentication>
    </pad-entry>
  </pad>
  <conn-entry>
     <name>nsf_h1-nsf_h2</name>
     <autostartup>start</autostartup>
     <version>ikev2</version>
```

```
        <initial-contact>false</initial-contact>
        <fragmentation>true</fragmentation>
        <ike-sa-lifetime-soft>
           <rekey-time>60</rekey-time>
           <reauth-time>120</reauth-time>
        </ike-sa-lifetime-soft>
        <ike-sa-lifetime-hard>
           <over-time>3600</over-time>
        </ike-sa-lifetime-hard>
        <authalg>7</authalg>
        <!--AUTH_HMAC_SHA1_160-->
        <encalg>3</encalg>
        <!--ENCR_3DES -->
        <dh-group>18</dh-group>
        <!--8192-bit MODP Group-->
        <half-open-ike-sa-timer>30</half-open-ike-sa-timer>
        <half-open-ike-sa-cookie-threshold>
           15
        </half-open-ike-sa-cookie-threshold>
        <local>
            <local-pad-entry-name>nsf_h1_pad</local-pad-entry-name>
        </local>
        <remote>
            <remote-pad-entry-name>nsf_h2_pad</remote-pad-entry-name>
        </remote>
        <spd>
          <spd-entry>
             <name>nsf_h1-nsf_h2</name>
             <ipsec-policy-config>
               <anti-replay-window>32</anti-replay-window>
               <traffic-selector>
                  <local-subnet>2001:DB8:1::0/64</local-subnet>
                  <remote-subnet>2001:DB8:2::0/64</remote-subnet>
                  <inner-protocol>any</inner-protocol>
                  <local-ports>
                    <start>0</start>
                    <end>0</end>
                  </local-ports>
                  <remote-ports>
                    <start>0</start>
                    <end>0</end>
                  </remote-ports>
               </traffic-selector>
               <processing-info>
                  <action>protect</action>
                  <ipsec-sa-cfg>
                     <pfp-flag>false</pfp-flag>
                     <ext-seq-num>true</ext-seq-num>
```

```
                     <seq-overflow>false</seq-overflow>
                     <stateful-frag-check>false</stateful-frag-check>
                     <mode>tunnel</mode>
                     <protocol-parameters>esp</protocol-parameters>
                     <esp-algorithms>
                        <!-- AUTH_HMAC_SHA1_96 -->
                        <integrity>2</integrity>
                        <!-- ENCR_AES_CBC -->
                        <encryption>12</encryption>
                        <tfc-pad>false</tfc-pad>
                     </esp-algorithms>
                     <tunnel>
                        <local>2001:DB8:123::100</local>
                        <remote>2001:DB8:123::200</remote>
                        <df-bit>clear</df-bit>
                        <bypass-dscp>true</bypass-dscp>
                        <ecn>false</ecn>
                    </tunnel>
                  </ipsec-sa-cfg>
               </processing-info>
             </ipsec-policy-config>
          </spd-entry>
        </spd>
        <child-sa-info>
           <!--8192-bit MODP Group -->
           <pfs-groups>18</pfs-groups>
           <child-sa-lifetime-soft>
              <bytes>1000000</bytes>
              <packets>1000</packets>
              <time>30</time>
              <idle>60</idle>
              <action>replace</action>
           </child-sa-lifetime-soft>
           <child-sa-lifetime-hard>
              <bytes>2000000</bytes>
              <packets>2000</packets>
              <time>60</time>
              <idle>120</idle>
           </child-sa-lifetime-hard>
        </child-sa-info>
      </conn-entry>
   </ipsec-ike>
```

Appendix E. Example of IKE-less case, transport mode (host-to-host).

This example shows a XML configuration file sent by the Security Controller to establish a IPsec Security association between two NSFs in transport mode (host-to-host) with ESP.

```
                             Security Controller
                                       |
                        /---- Southbound interface -----\
                       /                                  \
                      /                                    \
                     /                                      \
                    /                                        \
                 nsf_h1                                    nsf_h2
                 (:100)===== IPsec_ESP_Transport_mode =====(:200)
                               (2001:DB8:123:/64)
```

Figure 8: IKE-less case, transport mode.

```
<ipsec-ikeless
  xmlns="urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <spd>
    <spd-entry>
        <name>
           in/trans/2001:DB8:123::200/2001:DB8:123::100
        </name>
        <direction>inbound</direction>
        <reqid>1</reqid>
        <ipsec-policy-config>
           <traffic-selector>
             <local-subnet>2001:DB8:123::200/128</local-subnet>
             <remote-subnet>2001:DB8:123::100/128</remote-subnet>
             <inner-protocol>any</inner-protocol>
                <local-ports>
                   <start>0</start>
                   <end>0</end>
                </local-ports>
                <remote-ports>
                   <start>0</start>
                   <end>0</end>
                 </remote-ports>
           </traffic-selector>
           <processing-info>
              <action>protect</action>
              <ipsec-sa-cfg>
                <ext-seq-num>true</ext-seq-num>
```

```
                   <seq-overflow>true</seq-overflow>
                   <mode>transport</mode>
                   <protocol-parameters>esp</protocol-parameters>
                   <esp-algorithms>
                      <!--AUTH_HMAC_SHA1_96-->
                      <integrity>2</integrity>
                      <!--ENCR_AES_CBC -->
                      <encryption>12</encryption>
                   </esp-algorithms>
                 </ipsec-sa-cfg>
               </processing-info>
             </ipsec-policy-config>
           </spd-entry>
           <spd-entry>
             <name>out/trans/2001:DB8:123::100/2001:DB8:123::200</name>
             <direction>outbound</direction>
             <reqid>1</reqid>
             <ipsec-policy-config>
               <traffic-selector>
                 <local-subnet>2001:DB8:123::100/128</local-subnet>
                 <remote-subnet>2001:DB8:123::200/128</remote-subnet>
                 <inner-protocol>any</inner-protocol>
                 <local-ports>
                   <start>0</start>
                   <end>0</end>
                 </local-ports>
                 <remote-ports>
                   <start>0</start>
                   <end>0</end>
                 </remote-ports>
               </traffic-selector>
               <processing-info>
                 <action>protect</action>
                 <ipsec-sa-cfg>
                   <ext-seq-num>true</ext-seq-num>
                   <seq-overflow>true</seq-overflow>
                   <mode>transport</mode>
                   <protocol-parameters>esp</protocol-parameters>
                   <esp-algorithms>
                     <!-- AUTH_HMAC_SHA1_96 -->
                     <integrity>2</integrity>
                     <!-- ENCR_AES_CBC -->
                     <encryption>12</encryption>
                   </esp-algorithms>
                  </ipsec-sa-cfg>
                </processing-info>
              </ipsec-policy-config>
           </spd-entry>
```

```
        </spd>
        <sad>
          <sad-entry>
            <name>out/trans/2001:DB8:123::100/2001:DB8:123::200</name>
            <reqid>1</reqid>
            <ipsec-sa-config>
               <spi>34501</spi>
               <ext-seq-num>true</ext-seq-num>
               <seq-number-counter>100</seq-number-counter>
               <seq-overflow>true</seq-overflow>
               <anti-replay-window>32</anti-replay-window>
               <traffic-selector>
                 <local-subnet>2001:DB8:123::100/128</local-subnet>
                 <remote-subnet>2001:DB8:123::200/128</remote-subnet>
                    <inner-protocol>any</inner-protocol>
                    <local-ports>
                       <start>0</start>
                       <end>0</end>
                    </local-ports>
                    <remote-ports>
                       <start>0</start>
                       <end>0</end>
                    </remote-ports>
                </traffic-selector>
                <protocol-parameters>esp</protocol-parameters>
                <mode>transport</mode>
                <esp-sa>
                  <encryption>
                     <!-- //ENCR_AES_CBC -->
                     <encryption-algorithm>12</encryption-algorithm>
                     <key>01:23:45:67:89:AB:CE:DF</key>
                     <iv>01:23:45:67:89:AB:CE:DF</iv>
                  </encryption>
                  <integrity>
                     <!-- //AUTH_HMAC_SHA1_96 -->
                     <integrity-algorithm>2</integrity-algorithm>
                     <key>01:23:45:67:89:AB:CE:DF</key>
                  </integrity>
                </esp-sa>
            </ipsec-sa-config>
          </sad-entry>
          <sad-entry>
             <name>in/trans/2001:DB8:123::200/2001:DB8:123::100</name>
             <reqid>1</reqid>
             <ipsec-sa-config>
                 <spi>34502</spi>
                 <ext-seq-num>true</ext-seq-num>
                 <seq-number-counter>100</seq-number-counter>
```

```
                 <seq-overflow>true</seq-overflow>
                 <anti-replay-window>32</anti-replay-window>
                 <traffic-selector>
                    <local-subnet>2001:DB8:123::200/128</local-subnet>
                    <remote-subnet>2001:DB8:123::100/128</remote-subnet>
                    <inner-protocol>any</inner-protocol>
                    <local-ports>
                       <start>0</start>
                       <end>0</end>
                    </local-ports>
                    <remote-ports>
                       <start>0</start>
                       <end>0</end>
                    </remote-ports>
                 </traffic-selector>
                 <protocol-parameters>esp</protocol-parameters>
                 <mode>transport</mode>
                 <esp-sa>
                    <encryption>
                       <!-- //ENCR_AES_CBC -->
                       <encryption-algorithm>12</encryption-algorithm>
                       <key>01:23:45:67:89:AB:CE:DF</key>
                       <iv>01:23:45:67:89:AB:CE:DF</iv>
                    </encryption>
                    <integrity>
                       <!-- //AUTH_HMAC_SHA1_96 -->
                       <integrity-algorithm>2</integrity-algorithm>
                       <key>01:23:45:67:89:AB:CE:DF</key>
                    </integrity>
                  </esp-sa>
                  <sa-lifetime-hard>
                     <bytes>2000000</bytes>
                     <packets>2000</packets>
                     <time>60</time>
                     <idle>120</idle>
                  </sa-lifetime-hard>
                  <sa-lifetime-soft>
                     <bytes>1000000</bytes>
                     <packets>1000</packets>
                     <time>30</time>
                     <idle>60</idle>
                     <action>replace</action>
                  </sa-lifetime-soft>
            </ipsec-sa-config>
          </sad-entry>
       </sad>
   </ipsec-ikeless>
```

Appendix F. Examples of notifications.

Below we show several XML files that represent different types of notifications defined in the IKE-less YANG model, which are sent by the NSF to the Security Controller. The notifications happen in the IKE-less case.

```
<sadb-expire xmlns="urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless">
<ipsec-sa-name>in/trans/2001:DB8:123::200/2001:DB8:123::100
</ipsec-sa-name>
    <soft-lifetime-expire>true</soft-lifetime-expire>
       <lifetime-current>
          <bytes>1000000</bytes>
          <packets>1000</packets>
          <time>30</time>
          <idle>60</idle>
       </lifetime-current>
</sadb-expire>
```

Figure 9: Example of sadb-expire notification.

```
<sadb-acquire xmlns="urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless">
    <ipsec-policy-name>in/trans/2001:DB8:123::200/2001:DB8:123::100
    </ipsec-policy-name>
    <traffic-selector>
        <local-subnet>2001:DB8:123::200/128</local-subnet>
        <remote-subnet>2001:DB8:123::100/128</remote-subnet>
        <inner-protocol>any</inner-protocol>
         <local-ports>
              <start>0</start>
              <end>0</end>
         </local-ports>
         <remote-ports>
              <start>0</start>
              <end>0</end>
         </remote-ports>
    </traffic-selector>
</sadb-acquire>
```

Figure 10: Example of sadb-acquire notification.

```
   <sadb-seq-overflow
       xmlns="urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless">
         <ipsec-sa-name>in/trans/2001:DB8:123::200/2001:DB8:123::100
         </ipsec-sa-name>
   </sadb-seq-overflow>
```

Figure 11: Example of sadb-seq-overflow notification.

```
   <sadb-bad-spi
            xmlns="urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless">
           <spi>666</spi>
   </sadb-bad-spi>
```

Figure 12: Example of sadb-bad-spi notification.

Authors' Addresses

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 01
EMail: rafa@um.es

Gabriel Lopez-Millan
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 04
EMail: gabilm@um.es

Fernando Pereniguez-Garcia
University Defense Center
Spanish Air Force Academy, MDE-UPCT
San Javier (Murcia) 30720
Spain

Phone: +34 968 18 99 46
EMail: fernando.pereniguez@cud.upct.es